MATLAB基础与编程入门

张 威 编

西安电子科技大学出版社

2004

内 容 简 介

本书是学习 MATLAB 产品的最基础的入门书籍，重点介绍了 MATLAB 产品的体系，MATLAB 桌面工具的使用方法，M 语言的编程方法，MATLAB 进行数据可视化、分析处理的基本步骤以及部分常用的 MATLAB 工具，这些功能都是由 MATLAB 产品的核心——MATLAB 提供的，本书并没有涉及具体的产品工具箱。

书中不仅包含了 MATLAB 的基本使用方法，还包含了作者多年来使用 MATLAB 解决各种工程问题时积累的实际经验。该书内容翔实、全面、权威，示例丰富，不仅能够成为那些准备学习 MATLAB 软件的工程技术人员的入门书籍，也可以作为已经基本掌握 MATLAB 使用方法的工程技术人员学习、提高 MATLAB 使用技巧的参考书，同时，本书还可以作为 MATLAB 的培训课程教材。

图书在版编目(CIP)数据
MATLAB 基础与编程入门 / 张威编. —西安：西安电子科技大学出版社，2004.2
ISBN 7-5606-1330-6
Ⅰ. M…　Ⅱ. 张…　Ⅲ. 计算机辅助计算—软件包，MATLAB　Ⅳ. TP391.75
中国版本图书馆 CIP 数据核字(2003)第 113832 号

策　　划　毛红兵
责任编辑　毛红兵
出版发行　西安电子科技大学出版社（西安市太白南路 2 号）
电　　话　(029)88242885　88201467　　邮　编　710071
http: //www.xduph.com　　E-mail: xdupfxb@pub.xaonline.com
经　　销　新华书店
印刷单位　西安兰翔印刷厂
版　　次　2004 年 2 月第 1 版　2004 年 2 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印张 17
字　　数　403 千字
印　　数　1～4000 册
定　　价　23.00 元
ISBN 7-5606-1330-6/TP　0705
XDUP 1601001-1

前　言

MATLAB 软件最早成为商品化软件是在 1984 年，由 Mathworks 公司推出了 MATLAB 的第一个版本。到目前为止，MATLAB 的最新版本是 6.5.1，即 MATLAB Release 13 SP1。MATLAB 产品提供了大量丰富的应用函数，并且具有易扩充的开放性结构，在不断地吸收各行各业专家、学者、工程师的经验之后，成为国际上优秀的工程应用软件之一。目前，该软件包含 40 余个工具箱，30 余个专业仿真模块库，涵盖了控制系统应用、数字信号处理、数字图像处理、通讯、神经网络、小波理论分析、优化与统计、偏微分方程、动态系统实时仿真等多学科专业领域，其应用行业包括航空航天、汽车、通讯与 3G、医药卫生、化工工业、生物遗传工程、大学教育、国家政府机关以及金融财经等。在全球，MATLAB 的正式用户已经达到 60 余万，遍布世界上 100 多个国家和地区，自从上个世纪 90 年代 MATLAB 进入中国国内以来，MATLAB 软件在国内已经拥有了众多用户。目前国内很多高校已经在本科教学阶段将 MATLAB 作为一门必修课程，该软件已经成为控制、信号处理、通讯等领域研究生、本科生必须掌握的工具软件之一。

本书的组织如下：

第一章　对 MATLAB 产品体系进行介绍，并且介绍了 MATLAB 的桌面工具，这是学习 MATLAB 以及 MATLAB 其他产品的基础。

第二章　介绍 MATLAB 基本数据类型——矩阵和向量的创建方法。

第三章　介绍 MATLAB 中的各种数据类型，以及操作不同数据类型数据的函数。

第四章　介绍 M 语言编程的方法，并且介绍了在编程时经常使用的工具。

第五章　介绍 MATLAB 强大的数据 I/O 能力，以及不同数据文件的读写方法。

第六章　介绍 MATLAB 强大的数据可视化以及基本分析方法。

第七章　介绍利用 GUIDE 创建图形用户界面应用程序的方法。

另外本书的附录中还介绍了 MATLAB 的安装方法。

在本书的编写过程中，作者收集了国内外大量的最新的权威资料，结合 Mathworks 公司中国独家代理商——北京九州恒润科技有限公司多年来在 MATLAB 软件应用以及培训教学方面的经验，精心组织编写。书中不仅包含了 MATLAB 的基本使用方法，还包含了作者多年来使用 MATLAB 解决各种工程问题时积累的应用经验。该书内容翔实、全面、权威，示例丰富，不仅能够成为那些准备学习 MATLAB 软件的工程技术人员的入门书籍，也可以作为已经基本掌握 MATLAB 使用方法的工程技术人员学习、提高 MATLAB 使用技巧的参考书，同时，本书也可以作为 MATLAB 的培训课程教材。

如果中国国内的用户需要购买 MATLAB 软件，请按照下列地址与北京九州恒润科技有限公司联系：

■　公司总部

地址：北京市西城区北三环中路 27 号商房大厦 430 室

邮编：100029
电话：010-82011456
传真：010-62073600
■ 上海办事处
地址：上海市徐汇区漕宝路 70 号光大会展中心 D 座 505 室
邮编：200235
电话：021-64325413/5/6
传真：021-64325144
■ 成都办事处
地址：成都市人民南路一段 86 号城市之心大厦 23 楼 N 座
邮编：610016
电话：028-86203381/2/3
传真：028-86203381
北京九州恒润科技有限公司的互联网地址：**www.hirain.com**。
北京九州恒润科技有限公司的技术论坛：**www.hirain.com/forum/**

在本书的编写过程中，得到了西安电子科技大学出版社毛红兵编辑的大力支持，同时也得到了 **Mathworks** 公司中国独家代理商——北京九州恒润科技有限公司的鼎力协助，在这里对他们表示衷心的感谢。同时还要感谢父母和兄长多年对我的培养和教育，更要感谢我的女友——余志鸿小姐对我的关心和支持，正是有了他们的支持与鼓励才有了这本薄薄的小册子的出版。

由于时间仓促，书中难免存在一些不妥之处，诚望广大读者谅解，并且提出宝贵的意见和建议，以便我们在再版时改进。

如果需要得到本书所涉及的例子和练习的源代码，请直接与作者联系：
zhang_v@163.net

编著者
2003 年冬

目　录

第一章　概　　述

MATLAB 是一种流行的工程软件，可以应用于科学计算、控制系统设计与分析、数字信号处理、数字图像处理、通讯系统仿真与设计、金融财经系统分析等领域。在正式学习使用 MATLAB 之前，首先需要了解的就是 MATLAB 软件的基本环境及其使用方法。本章将简要介绍一下 MATLAB 软件产品的体系，重点介绍 MATLAB 软件的图形界面环境的基本使用方法。

本章讲述的主要内容如下：

- □□□□□□产品族简介；
- □□□□□□的桌面环境；
- □□□□□□用户界面窗口的使用。

1.1　MATLAB 产品族简介

MATLAB 的名称源自 Matrix Laboratory，它的首创者是在数值线性代数领域颇有影响的 Cleve Moler 博士，他也是生产经营 MATLAB 产品的美国 Mathworks 公司的创始人之一。MATLAB 是一种科学计算软件，专门以矩阵的形式处理数据。MATLAB 将高性能的数值计算和可视化集成在一起，并提供了大量的内置函数，从而使其被广泛地应用于科学计算、控制系统、信息处理等领域的分析、仿真和设计工作中，而且利用 MATLAB 产品的开放式结构，用户可以非常容易地对 MATLAB 的功能进行扩充，从而在不断深化对问题认识的同时，逐步完善 MATLAB 产品以提高产品自身的竞争能力。

MATLAB 产品族可以用来进行如下工作：

- 数值分析；
- 数值和符号计算；
- 工程与科学绘图；
- 控制系统的设计与仿真；
- 数字图像处理；
- 数字信号处理；
- 通讯系统设计与仿真；
- 财务与金融工程。

目前，MATLAB 的最新版本为 MATLAB 6.5.1，Mathworks 公司将其称之为 MATLAB Release 13 Service Pack 1(R13 SP1)，本书就以该版本的 MATLAB 为基础制作完成。

1.1.1　MATLAB 的产品体系

MATLAB 产品由若干个模块组成，不同的模块完成不同的功能，其中有

- MATLAB；
- MATLAB Toolboxes；
- MATLAB Compiler；
- Simulink；
- Simulink Blockset；
- Real-Time Workshop (RTW)；
- Stateflow；
- Stateflow Coder。

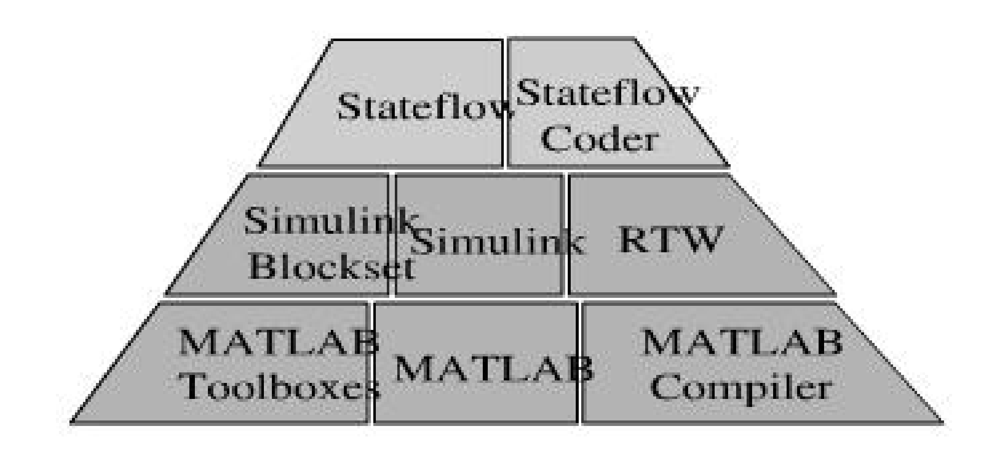


图 1-1　MATLAB 的产品体系

由这些模块构成的 MATLAB 产品体系如图 1-1 所示。

其中，MATLAB 是 MATLAB 产品家族的基础，它提供了基本的数学算法，例如矩阵运算、数值分析算法，MATLAB 集成了 2D 和 3D 图形功能，以完成相应数值可视化的工作，并且提供了一种交互式的高级编程语言——M 语言，利用 M 语言可以通过编写脚本或者函数文件实现用户自己的算法。

MATLAB Compiler 是一种编译工具，它能够将那些利用 MATLAB 提供的编程语言——M 语言编写的函数文件编译生成标准的 C/C++语言源文件，而生成的标准 C/C++源代码可以被任何一种 C/C++编译器编译生成函数库或者可执行文件，这样就可以扩展 MATLAB 功能，使 MATLAB 能够同其他高级编程语言(例如 C/C++语言)进行混合应用，取长补短，以提高程序的运行效率，丰富程序开发的手段。

MATLAB 除了能够和 C/C++语言集成开发以外，目前的 MATLAB 还提供了和 Java 语言接口的能力，并且它还支持 COM 标准，能够和任何一种支持 COM 标准的软件协同工作。另外，在 Release 13 中，包含了 MATLAB Compiler 的扩展产品——MATLAB COM Builder 和 Excel Builder，分别用来将 MATLAB 的函数文件打包成 COM 组件或者 Excel 插件，将 MATLAB 应用程序算法集成到相应的开发工具或者应用软件中。

利用 M 语言还开发了相应的 MATLAB 专业工具箱函数供用户直接使用，这些工具箱应用的算法是开放的、可扩展的，用户不仅可以察看其中的算法，还可以针对一些算法进行修改，甚至允许开发自己的算法以便扩充工具箱的功能。目前 MATLAB 产品的工具箱有 40 多种，分别涵盖了数据获取、科学计算、控制系统设计与分析、数字信号处理、数字图像处理、金融财务分析以及生物遗传工程等专业领域。

MATLAB 主要的专业工具箱包括以下几种。

- 数学与数据分析：

  Optimization
  Statics
  Nerual Network
  Symbolic Math

Partial Differential Equation
Mapping
Spline
Curve Fitting
Virtual Reality
Bioinforamtics

- 数据获取与采集：

Data Acquisition
Image Acquisition
Instrument Control
Database
Excel

- 信号处理与图像处理：

Signal Processing
Image Processing
Communication
System Identification
Wavelet
Filter Design
MATLAB Link for Code Composer Studio

- 控制系统设计与分析：

Control system
Fuzzy Logic
Robust Control16
Mu-Analysis and Synthesis18
LMI Control18
Model Predictive Control18
Model-Based Calibration

- 财经与金融：

Financial
Financial Time Series
GARCH
Datafeed
Financial Derivatives
Fixed Income

本书中所介绍的内容不包括上述若干工具箱，本书将集中介绍 MATLAB 基本模块的使用方法，有关产品工具箱的介绍请参阅 MATLAB 的帮助文档信息。图 1-1 中提及的其他产品将在下面的几个小节分别介绍。

1.1.2　Simulink 简介

Simulink 是基于 MATLAB 的框图设计环境，可以用来对各种动态系统进行建模、分析和仿真，它的建模范围广泛，可以针对任何能够用数学来描述的系统进行建模，例如航空航天动力学系统、卫星控制制导系统、通讯系统、船舶及汽车等，其中包括连续、离散，条件执行、事件驱动、单速率、多速率和混杂系统等。Simulink 提供了利用鼠标拖放的方法建立系统框图模型的图形界面，而且 Simulink 还提供了丰富的功能块以及不同的专业模块集合，利用 Simulink 几乎可以做到不书写一行代码就能完成整个动态系统的建模工作。

此外，在 Simulink 基础上还提供了 Stateflow，用来进行事件驱动过程的仿真。

Simulink 的特点：

- 交互式建模：Simulink 本身就提供了大量的功能块方便用户快速建立动态系统的模型，如图 1-2 所示，建模的时候只需要利用鼠标拖放功能块并将其连接起来即可。

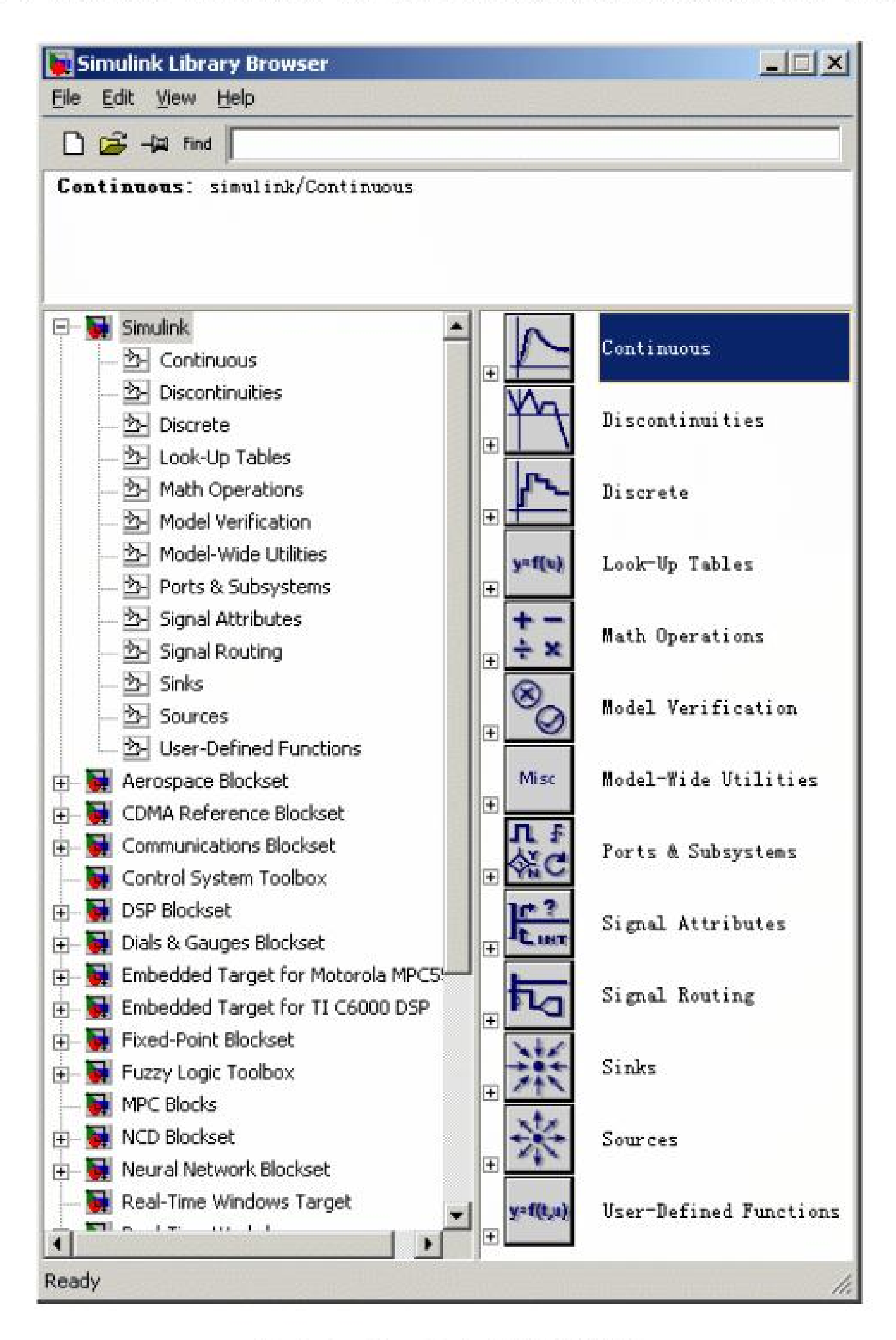


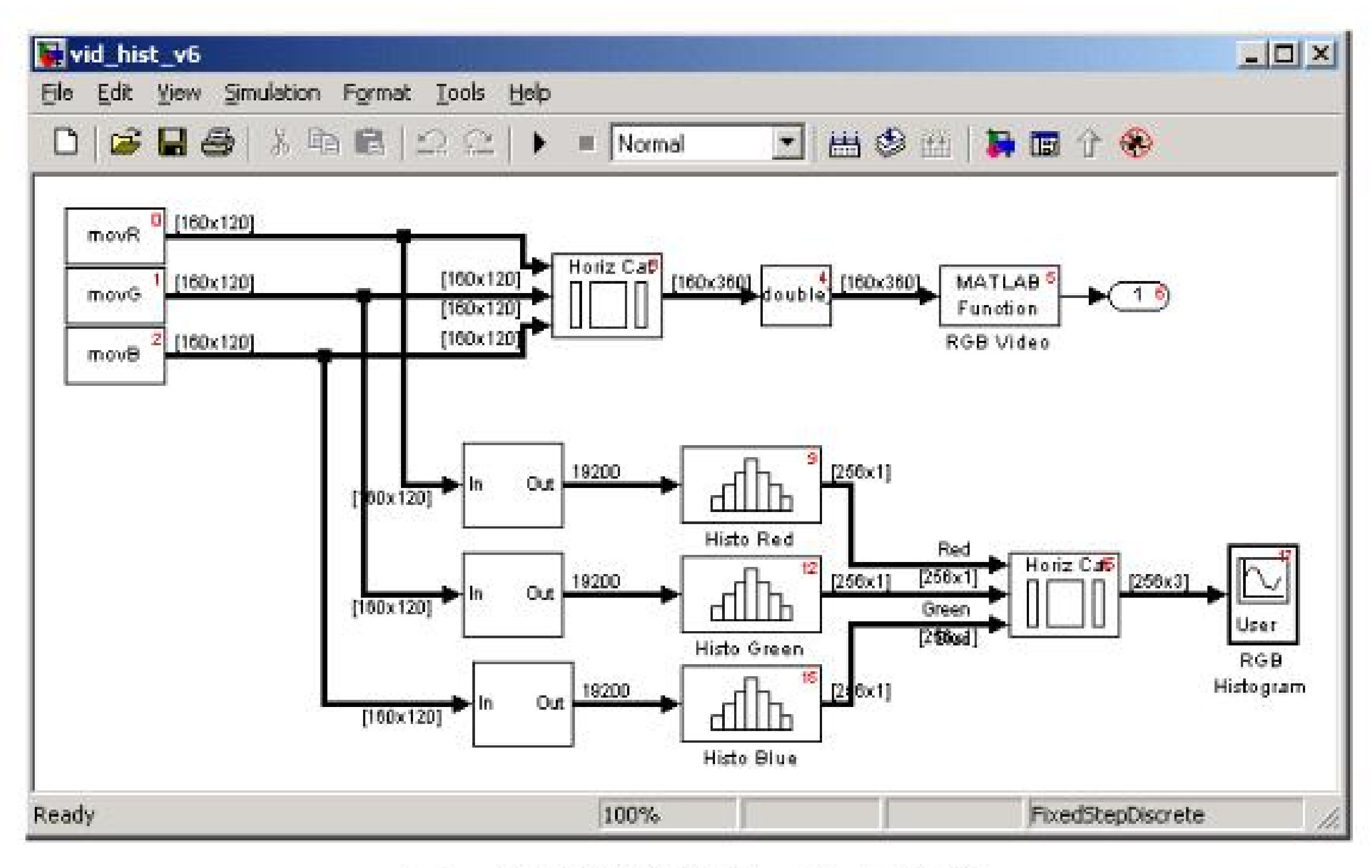
图 1-2　Simulink 的库浏览器

- 交互式仿真：Simulink 的框图提供可交互的仿真环境，可以将仿真结果动态显示出来，并且在各种仿真的过程中调节系统的参数。
- 任意扩充和定制功能：Simulink 的开放式结构允许用户扩充仿真环境的功能，可以将用户利用 C、C++、Fortran 语言编写的算法集成到 Simulink 框图中。
- 与 MATLAB 工具集成：Simulink 的基础是 MATLAB，在 Simulink 框图中可以直接利用 MATLAB 的数学、图形和编辑功能，完成诸如数据分析、过程自动化分析、优化参数等工作。
- 专业模型库：为了扩展 Simulink 的功能，Mathworks 公司针对不同的专业领域和行业开发了各种专业模型库，将这些模型库同 Simulink 的基本模块库结合起来，可以完成不同专业领域动态系统的建模工作。Simulink 的相关产品以及专业模块如表 1-1 所示。

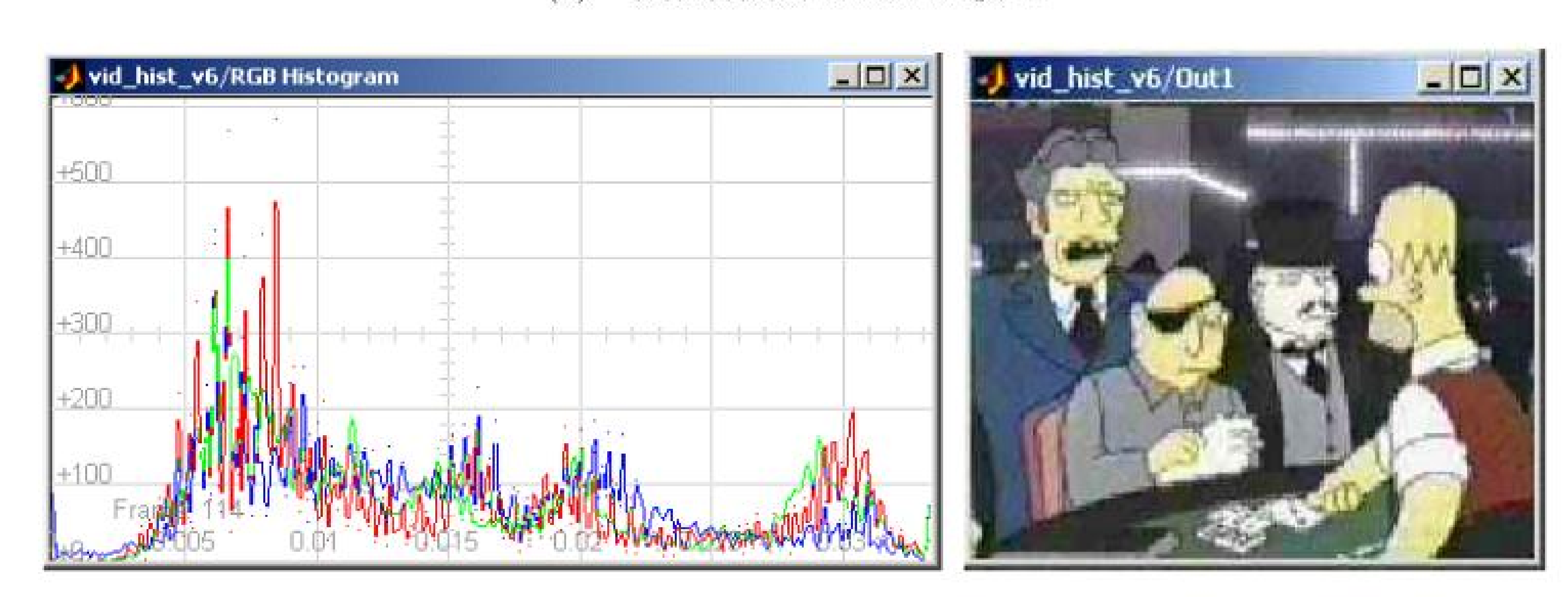
表 1-1　**MATLAB 中的 Simulink 相关模块**

产 品 名 称	描　述
Simulink	图形化建模仿真环境
Simulink Performance Tools	性能优化调试工具
Simulink Report Generator	Simulink 自动文档生成工具
Stateflow	基于事件驱动的建模工具
Stateflow Coder	Stateflow 的代码生成工具
Real-Time Workshop	实时代码生成工具
Real-Time Embedded Coder	嵌入式实时代码生成选项
Real-Time Windows Target	Windows 环境下的实时代码生成选项
xPC Target	基于 xPC 的实时代码生成选项
xPC Target Embedded Option	嵌入式 xPC 实时代码选项
Embedded Target for Motorola MPC555	Motorola MPC555 的嵌入式代码生成选项
Embedded Target for TI C6000 DSP	TI C6000 系列的嵌入式代码生成选项
Embedded Target for Infineon C166	Infineon C166 的嵌入式代码生成选项
Embedded Target for Motorola HC12	HC12 的嵌入式代码生成选项
Embedded Target for OSEK/VDX	OSEK 操作系统的嵌入式代码生成选项
Aerospace Blockset	航空航天及国防专业模块库
Fixed-Pointed Blockset	定点代码应用专业模块库
DSP Blockset	数字信号处理系统专业模块库
Communication Blockset	通讯系统仿真专业模块库
CDMA Blockset	IS95A 通讯系统仿真专业模块库
Dials & Gauges Blockset	虚拟仪器仪表专业模块库
Nonlinear Control Design Blockset	非线性控制设计专业模块库
SimMechanics	机械系统仿真专业模块库
SimPowerSystem	电力电子系统仿真专业模块库
Virtual Reality Toolbox	虚拟现实应用工具箱

Simulink 不仅可以建立控制系统的动态模型，而且还能够创建数字信号处理系统，甚至视频系统的动态模型，利用 Simulink 提供的各种工具，特别是实时代码生成工具，可以完成相应的代码验证工作，图 1-3(a)、(b)为 Simulink 进行视频系统仿真的例子。



(a)　视频仿真的 Simulink 模型



(b)　视频仿真的运行结果

图 1-3　用 Simalink 进行视频系统仿真的例子

可以利用 MATLAB 产品中提供的 Embedded Target for TI C6000 DSP 产品将该模型下载到 6416 DSK 开发板中完成算法的验证与测试。

1.1.3　Stateflow 简介

Stateflow 是一个交互式的设计工具，它基于有限状态机的理论，可以用来对复杂的事件驱动系统进行建模和仿真。Stateflow 与 Simulink 和 MATLAB 紧密集成，可以将 Stateflow 创建的复杂控制逻辑有效地结合到 Simulink 的模型中。

有限状态机是具有有限个状态的系统的理论表述。它以某些缺省的状态为起点，根据所定义的事件和转移进行操作，转移表示状态机如何对事件进行响应(控制流程)。

图 1-4 就是有限状态机的一个例子。其中，A、B、C、D、E 分别表示系统的不同状态，而 a、b 表示响应的事件，具有方向的线表示状态与状态之间的逻辑流，逻辑流依赖事件驱动，所以这是一个典型的事件驱动模型，可利用有限状态机理论进行表述。

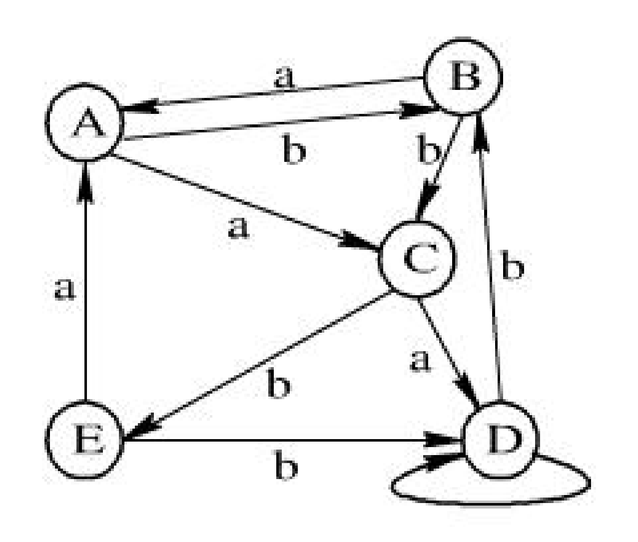


图 1-4　有限状态机的一个例子

假设用一个状态机表示空调的制冷工作，一般空调工作的时候具有两种状态，即运行(On)和停止(Off)。当电源接通之后，空调机一般默认为运行状态，若室内的温度高于设定的温度，则空调机处于运行状态，若室内的温度低于设定的温度，则空调机停止运行，这样系统就从一个状态转换到另一个状态。利用 Stateflow 可以对该系统进行建模，如图 1-5 所示。

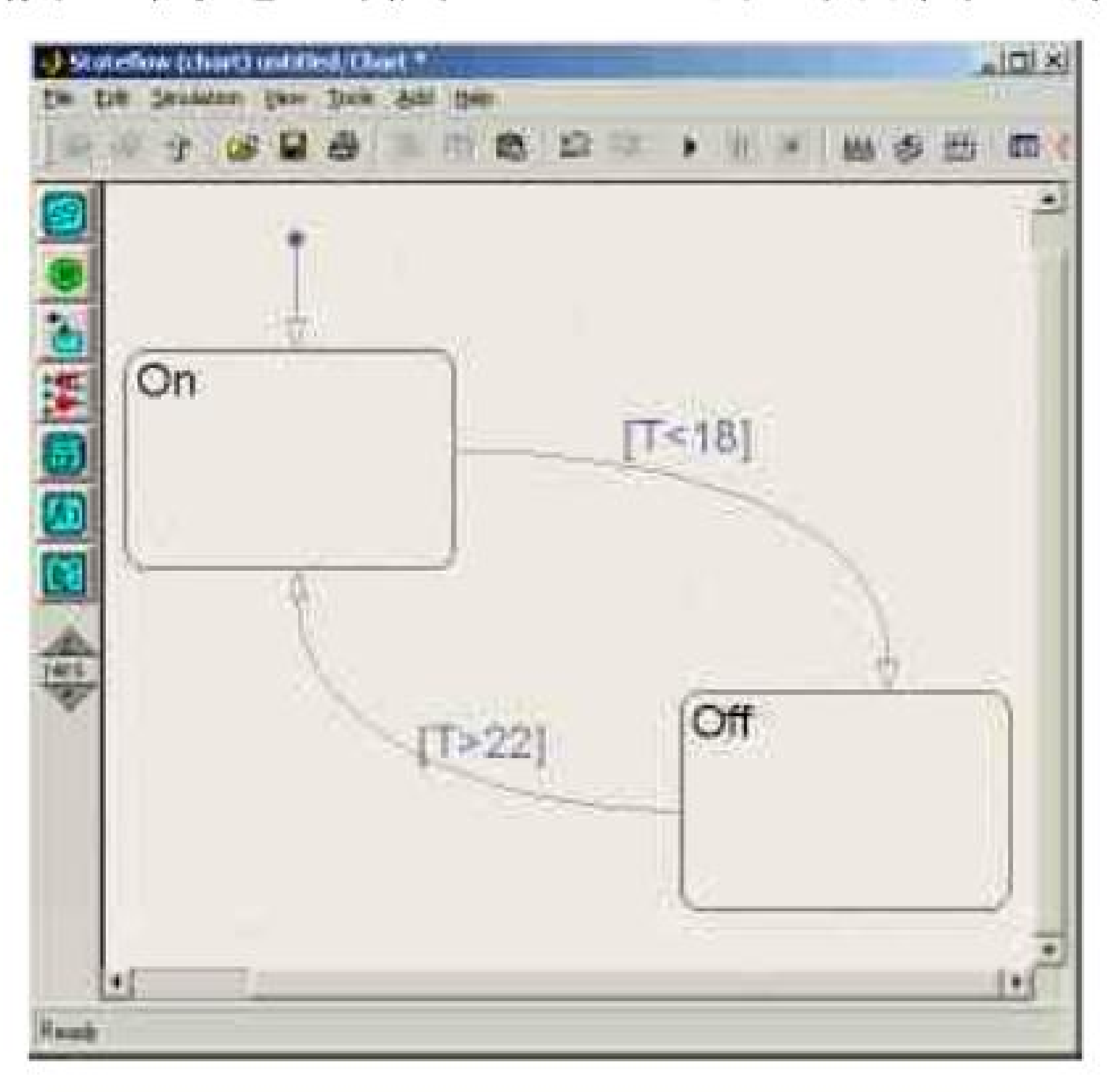


图 1-5　利用 Stateflow 建模

Stateflow 主要可以用于各种动态逻辑、控制流程系统的建模与仿真，例如在飞行器的导航制导与控制系统中，经常需要根据当前的飞行状态切换不同的系统控制参数，利用 Stateflow 就可以完成此类系统的建模与仿真。再比如说，在通讯系统中，为了仿真网络通讯中的物理层(MAC Layer)协议，也可以使用 Stateflow 进行建模与仿真，图 1-6 展示了用于仿真以太网物理层协议的 Stateflow 模型。

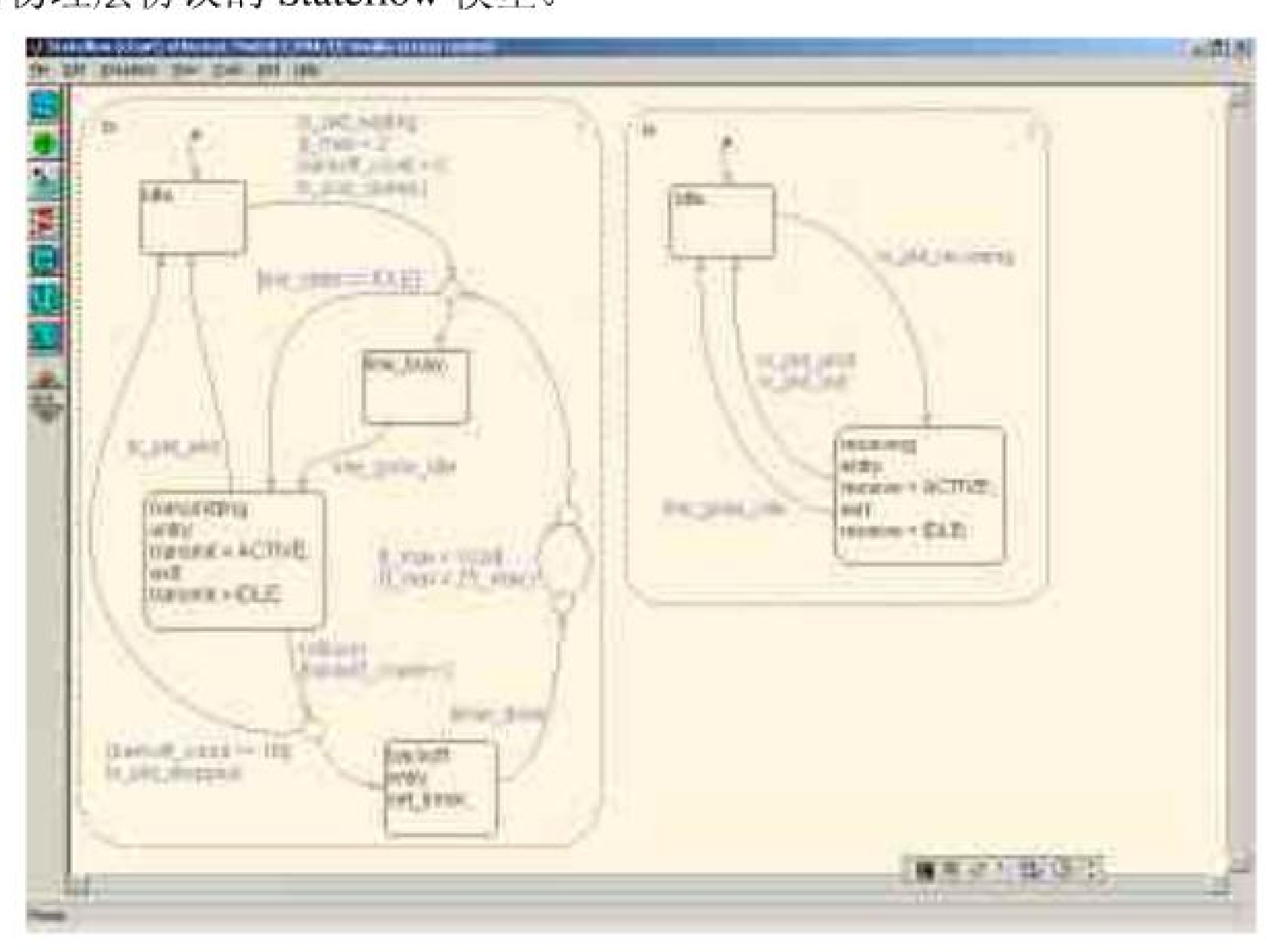


图 1-6　利用 Stateflow 完成以太网物理层协议仿真

1.1.4 自动化代码生成工具

在 MATLAB 产品族中，自动化的代码生成工具主要有 Real-Time Workshop(RTW)和 Stateflow Coder，这两种代码生成工具可以直接将 Simulink 的模型框图和 Stateflow 的状态图转换成高效、优化的程序代码。利用 RTW 生成的代码简洁、可靠、易读。目前 RTW 支持生成标准的 C 语言代码，并且具备了生成其他语言代码的能力。整个代码的生成、编译以及相应的目标下载过程都是自动完成的，用户需要做的仅仅是使用鼠标点击几个按钮即可。Mathworks 公司针对不同的实时或非实时操作系统平台开发了相应的目标选项，以配合不同的软、硬件系统完成快速控制原型(Rapid Control Prototype)开发、硬件在回路的实时仿真(Hardware-in-Loop)、产品代码生成等工作。

Read-Time Workshop 的体系结构如图 1-7 所示。

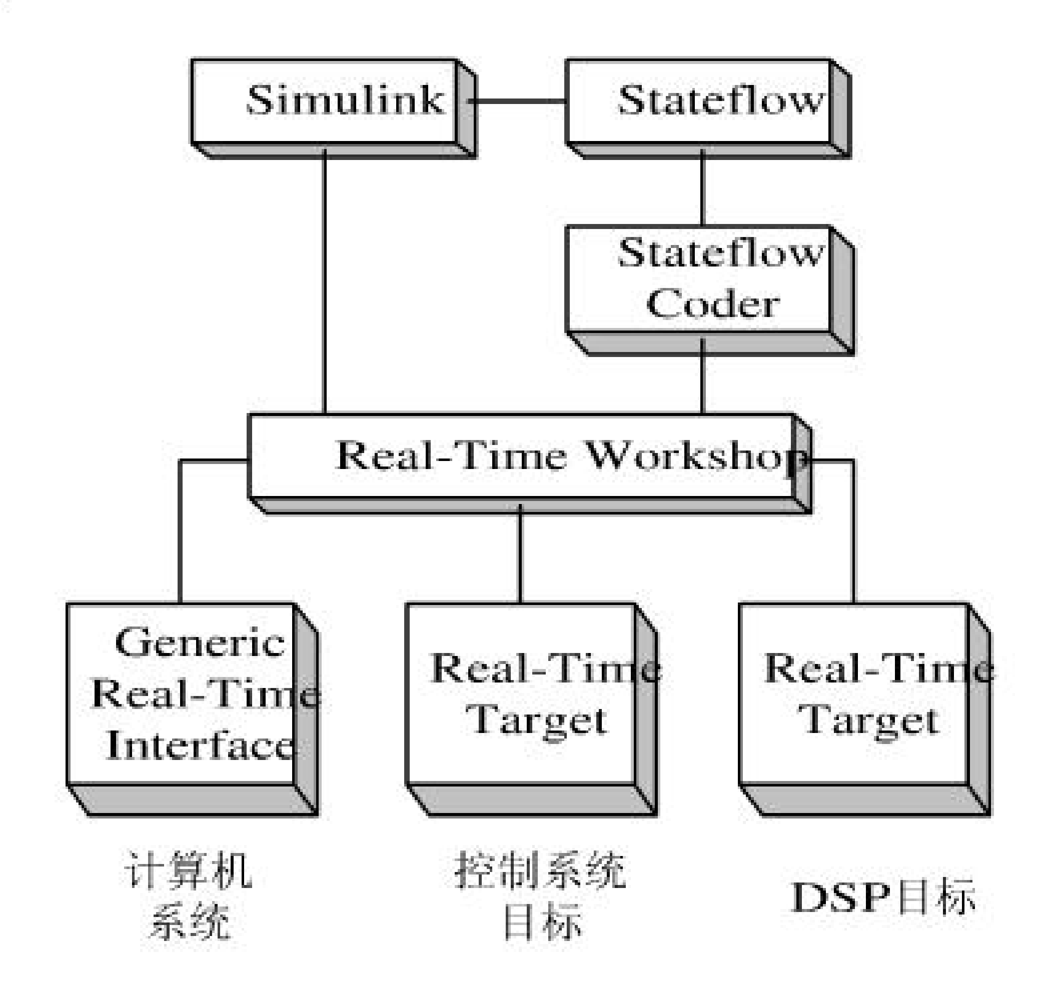


图 1-7　Real-Time Workshop 的体系结构

目前 MATLAB/Simulink 的 RTW 支持的目标主要包括：

- S-function Target；
- Rapid Simulation Target；
- General Real-Time Target；
- DOS(4GW)Real-Time Target；
- Real-Time Windows Target；
- LE/O Real-Time Target；
- Tornado(VxWorks)Real-Time Target；
- Embedded Target for Motorola MPC555；
- Embedded Target for TI TMS320 C6000 DSP；
- Embedded Target for Infineon C166；
- Embedded Target for Motorola HC12；
- Embedded Target for OSEK/VDX；
- xPC Target；

■ 第三方硬件平台。

在这些目标体系中，常用于控制系统原型仿真的目标平台主要包括 Real-Time Windows Target、xPC Target 以及 dSPACE 等第三方硬件平台。

Real-Time Windows Target 可以将一台运行 Windows 操作系统的 PC 机转变成为独立的自主目标机，目标机交互地实时运行 Simulink 模型。Real-Time Windows Target 支持直接的 I/O 访问，允许用户和模型之间实时交互，是一个易用的、廉价的低端快速原型开发和硬件在回路仿真的目标环境。

利用 xPC Target，用户可以将 Simulink 提供的支持 xPC Target 的 I/O 模块添加到 Simulink 的模型中，然后通过 RTW 的 xPC Target 选项，将模型下载到另一台运行在 xPC Target 实时内核的 x86 体系计算机上，这时模型在 xPC 实时内核上实时地运行。xPC 支持的板卡和 I/O 数量种类丰富，是理想的快速原型实现和硬件在回路仿真的测试工具。

在第三方硬件支持中，目前最流行的也是性能较高的就是德国 dSPACE 公司开发的 dSPACE 系统。dSPACE 系统是一套基于 MATLAB/Simulink 的控制系统开发及测试平台，实现了与 MATLAB/Simulink 的无缝连接。在 MATLAB/Simulink 和 dSPACE 系统的支持下，可以完善地解决控制系统的快速原型开发和硬件在回路仿真中遇到的各种问题，使工程师可以将自己的精力集中在控制系统算法的开发中，而不是耗费在大量繁琐的代码编写中。

dSPACE 系统的主要特点如下：

- ■ 组合性强：dSPACE 系统设计了各种标准的组件系统，可以对系统进行任意的组合和配置，可以选择不同的处理器板卡，也可以针对具体的应用选择性能不同的 I/O 板卡。
- ■ 易于掌握：由于 dSPACE 系统和 MATLAB/Simulink 产品之间的无缝连接，使掌握了 MATLAB 的广大工程技术人员可以轻松掌握 dSPACE 系统。
- ■ 快速性好：由于 dSPACE 系统和 MATLAB/Simulink 产品之间的无缝连接，整个生成代码、编译、下载过程由计算机自动完成，而修改参数再次生成代码、编译、下载只需要几分钟，从而可以在短时间内对原型进行反复的更改和试验，减少了以往大量的编码修改、编写的时间，极大地提高了工作效率。

此外，dSPACE 系统还具有较高的实时性、可靠性，而且通过不同的板卡配置提供了灵活多变的应用，从而适应用户的各方面要求。

关于 dSPACE 系统的详细信息, 请浏览该公司的网站：http://www.dspaceinc.com, 也可以浏览该产品在中国的惟一代理商——北京九州恒润科技有限公司的网站：http://www.hirain.com。

像各种单片机一样，HC12、C166 等主要用于控制系统中嵌入式控制器的处理器在回路中的仿真开发，特别是 Embedded Target for TI C6000 DSP 能够将 Simulink 框图化的模型转变成直接在 C6000 系列 DSP 开发板或者 EVM 板上运行的可执行程序，从而直接通过硬件设备来验证系统的算法。

另外，MATLAB 开放性的可扩充体系允许用户开发自定义的系统目标，利用 Real-Time Workshop Embedded Coder 能够直接将 Simulink 的模型转变成效率优化的产品级代码，代码不仅可以是浮点的，还可以是定点的。

1.2　MATLAB 的桌面环境

在运行 MATLAB 之前首先要在自己的操作系统中安装 MATLAB，目前 MATLAB 可以在 Windows、Red-hat Linux、Sun Solaris、MAC OS 等操作系统中安装使用。如果读者使用 Windows 操作系统，则建议使用 Windows 2000 或者 Windows XP Professional 版本作为 MATLAB 的运行平台。运行 MATLAB 时，可以双击 MATLAB 的图标，或者在命令行提示符(控制台方式)下键入指令：matlab，这时将启动 MATLAB 的图形桌面工具环境。

MATLAB 的桌面环境可以包含多个窗口，这些窗口分别为历史命令窗口(Command History)、命令行窗口(Command Window)、当前目录浏览器(Current Directory Browser)、工作空间浏览器(Workspace Browser)、目录分类窗口(Launch Pad)、数组编辑器(Array Editor)、M 文件编辑器/调试器(Editor/Debugger)、超文本帮助浏览器(Help Navigator/Browser)，这些窗口都可以内嵌在 MATLAB 主窗体中，组成 MATLAB 的用户界面。其中当 MATLAB 安装完毕并首次运行时，展示在用户面前的界面为 MATLAB 运行时的缺省界面窗口，如图 1-8 所示。

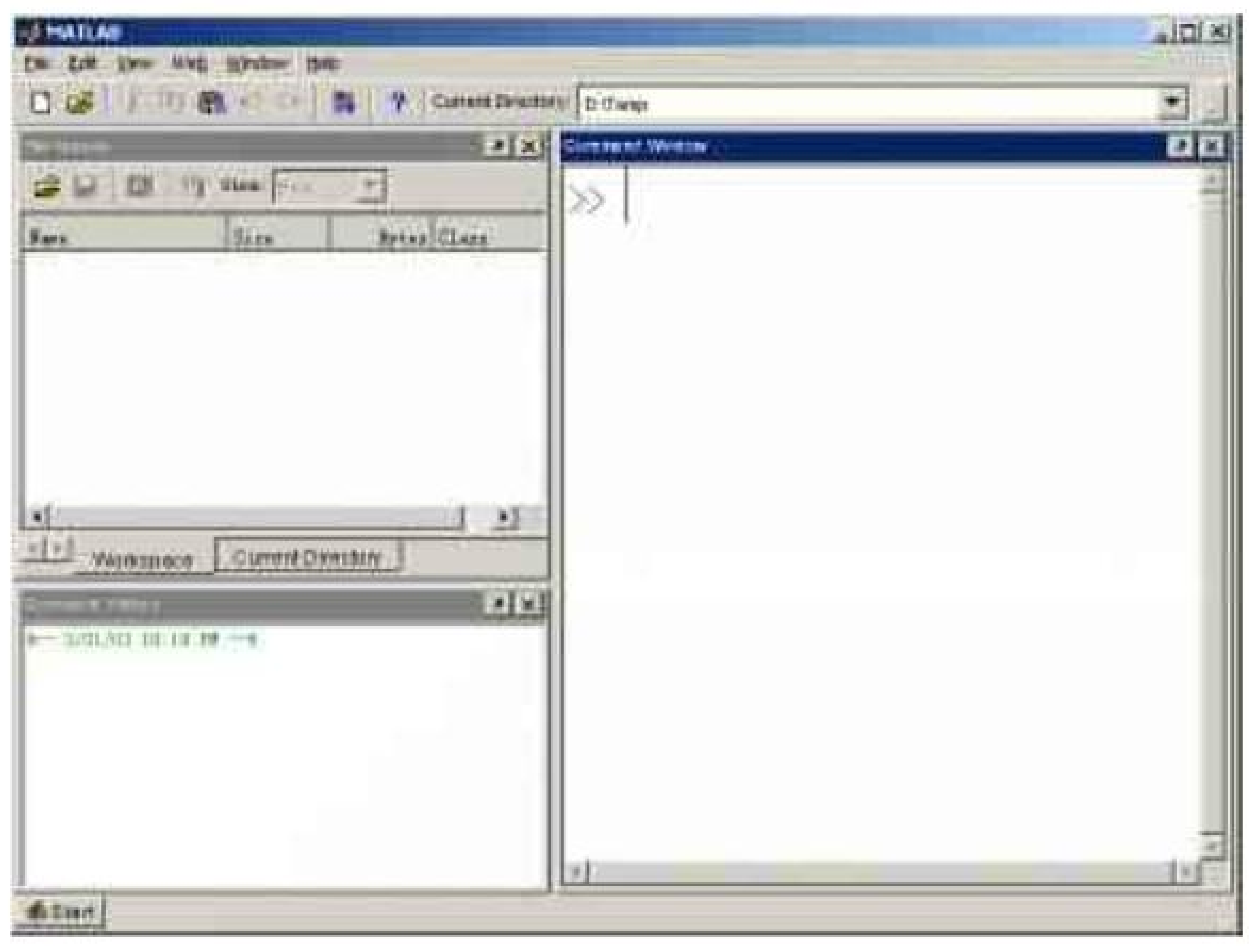


图 1-8　MATLAB 默认的用户界面

MATLAB 6.5 的缺省界面和 MATLAB 6.1 的缺省界面相比还是有一定变化的，主要就是在界面的左下角有一个名为“Start”的启动菜单，在这个菜单中可以执行 MATLAB 产品的各种工具，并且可以查阅 MATLAB 包含的各种资源，它的功能和 MATLAB 的目录分类窗口(Launch Pad)功能非常类似。在默认的缺省用户界面中，新版本的 MATLAB 中没有了目录分类窗口(Launch Pad)，而是工作空间浏览器(Workspace Browser)。

MATLAB 启动的界面可以具有多种默认的选择，用户可以通过 MATLAB 界面中的“View”菜单下的“Desktop Layout”子菜单下的命令选择不同的 MATLAB 界面，这些命

令分别为：

- Default:缺省的界面，如图 1-8 所示。包含历史命令窗口(Command History)、命令行窗口(Command Window)，此外工作空间浏览器(Workspace Browser)和当前目录浏览器(Current Directory Browser)两个窗口层叠在一起。
- Command Windows Only：仅包含命令行窗口(Command Window)，此时 MATLAB 界面的外观类似于旧版本的 MATLAB。
- Simple:包含两个窗口——命令行窗口(Command Window)和历史命令窗口(Command History)，两个窗口并列在界面中，如图 1-9 所示。

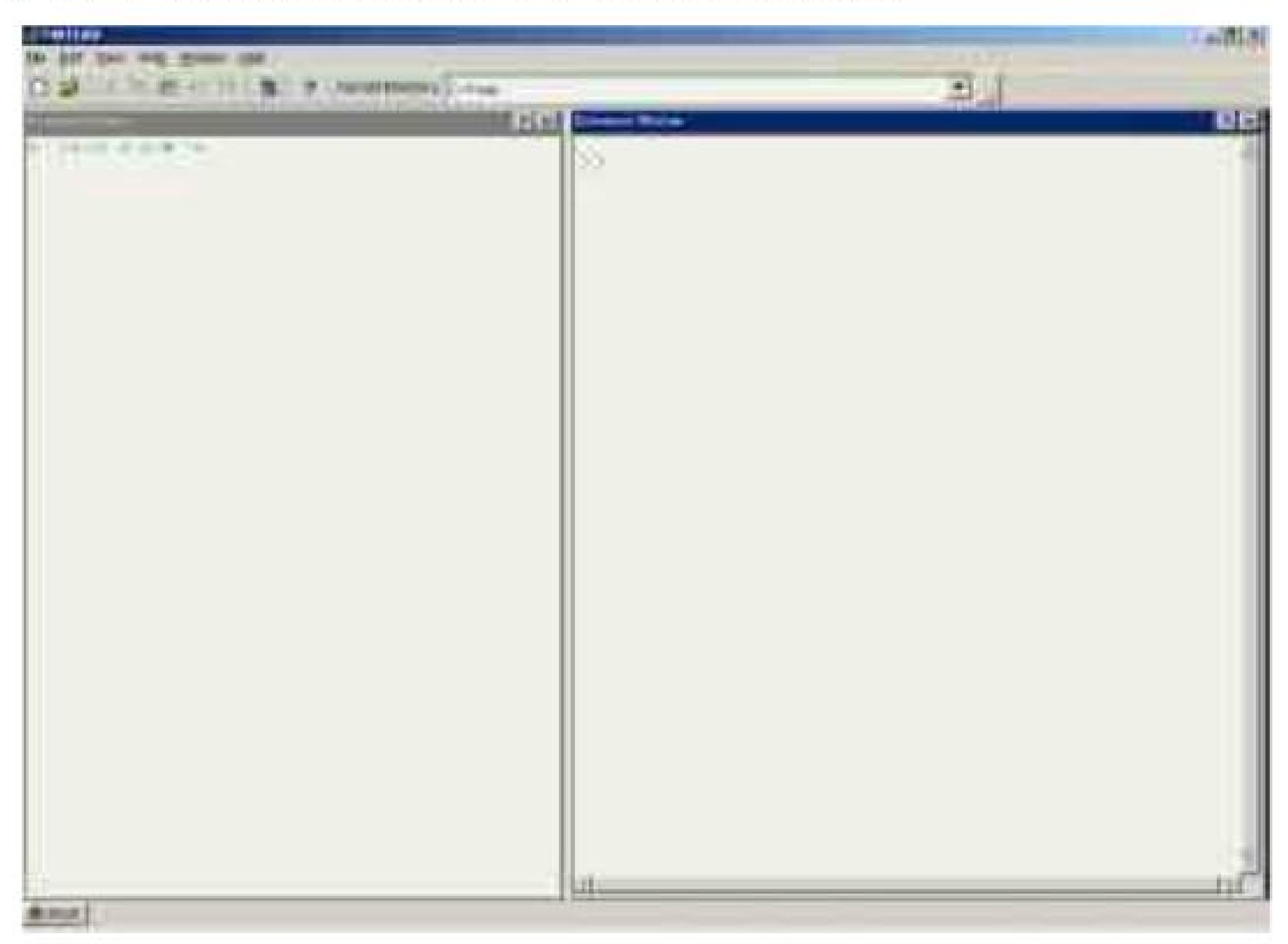


图 1-9　选择 Simple 菜单命令的界面

- Short History 和 Tall History：这两个菜单命令包含的窗口类型和数量同默认的界面完全一致，不过排放的顺序不同，如图 1-10、1-11 所示。

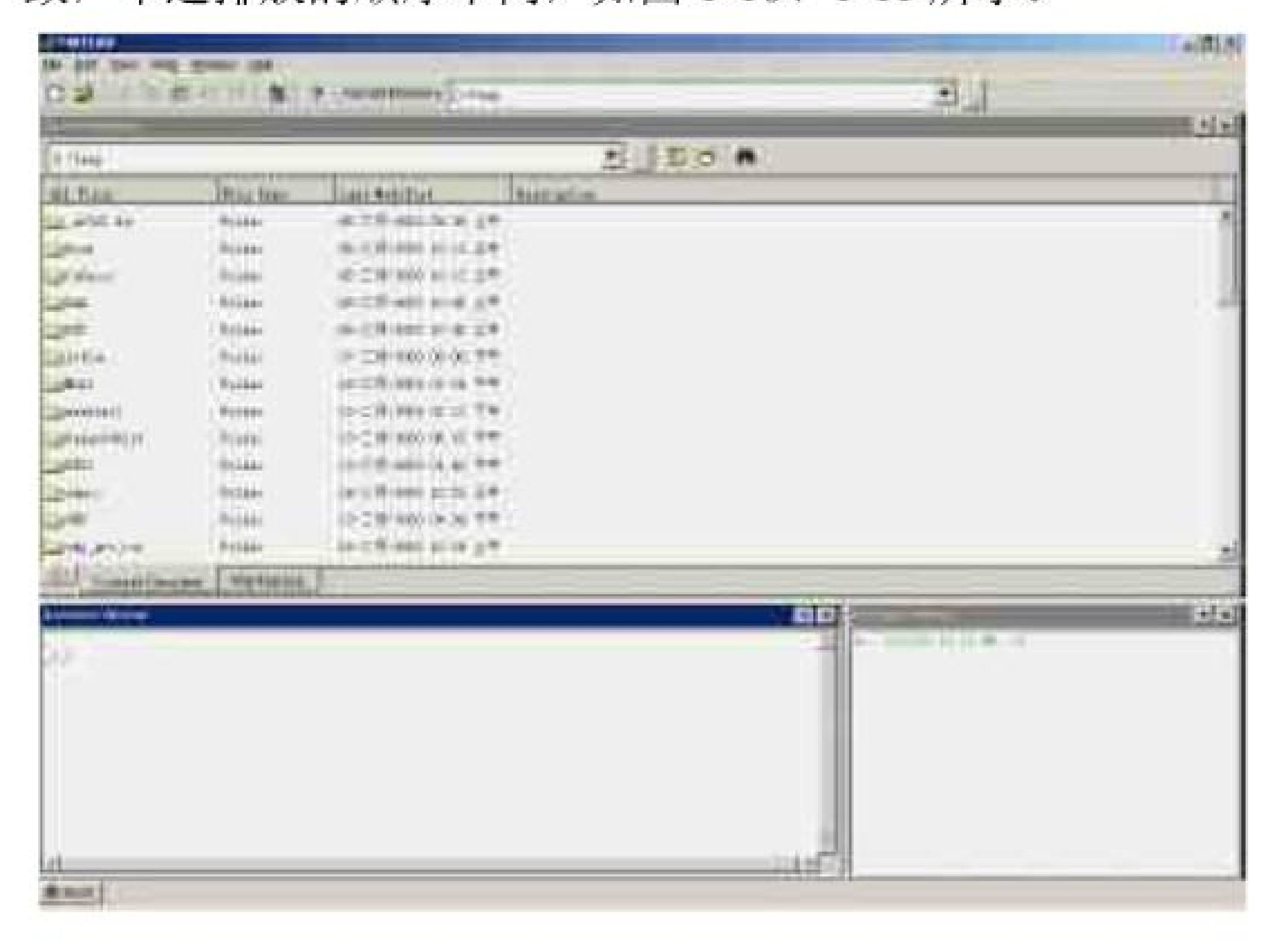


图 1-10　选择 Short History 菜单命令的界面

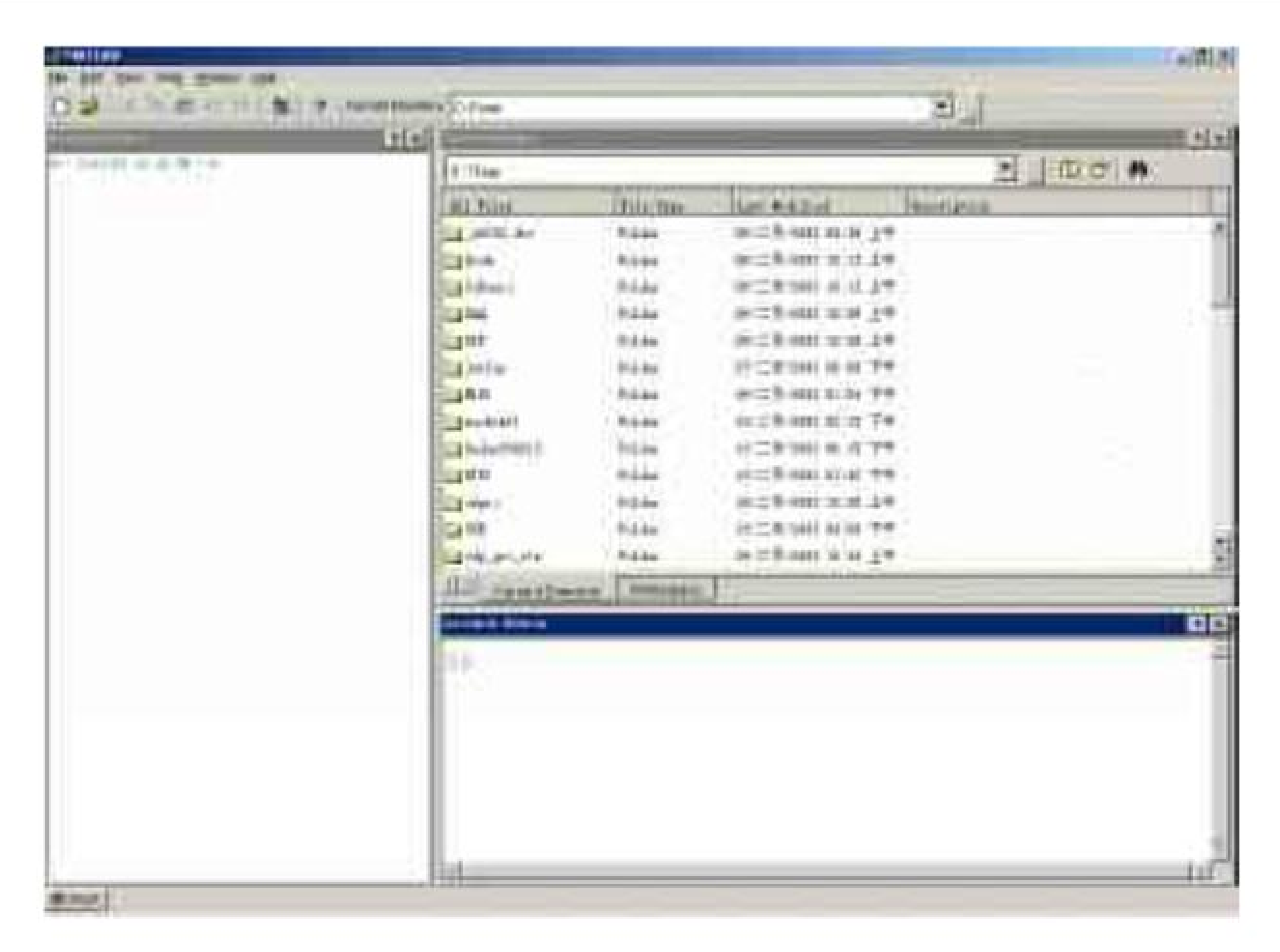


图 1-11　选择 Tall History 菜单命令的界面

■ Five Panel：包含所有的 MATLAB 桌面窗口，在 MATLAB 界面中各个窗口处于平铺状态，如图 1-12 所示。

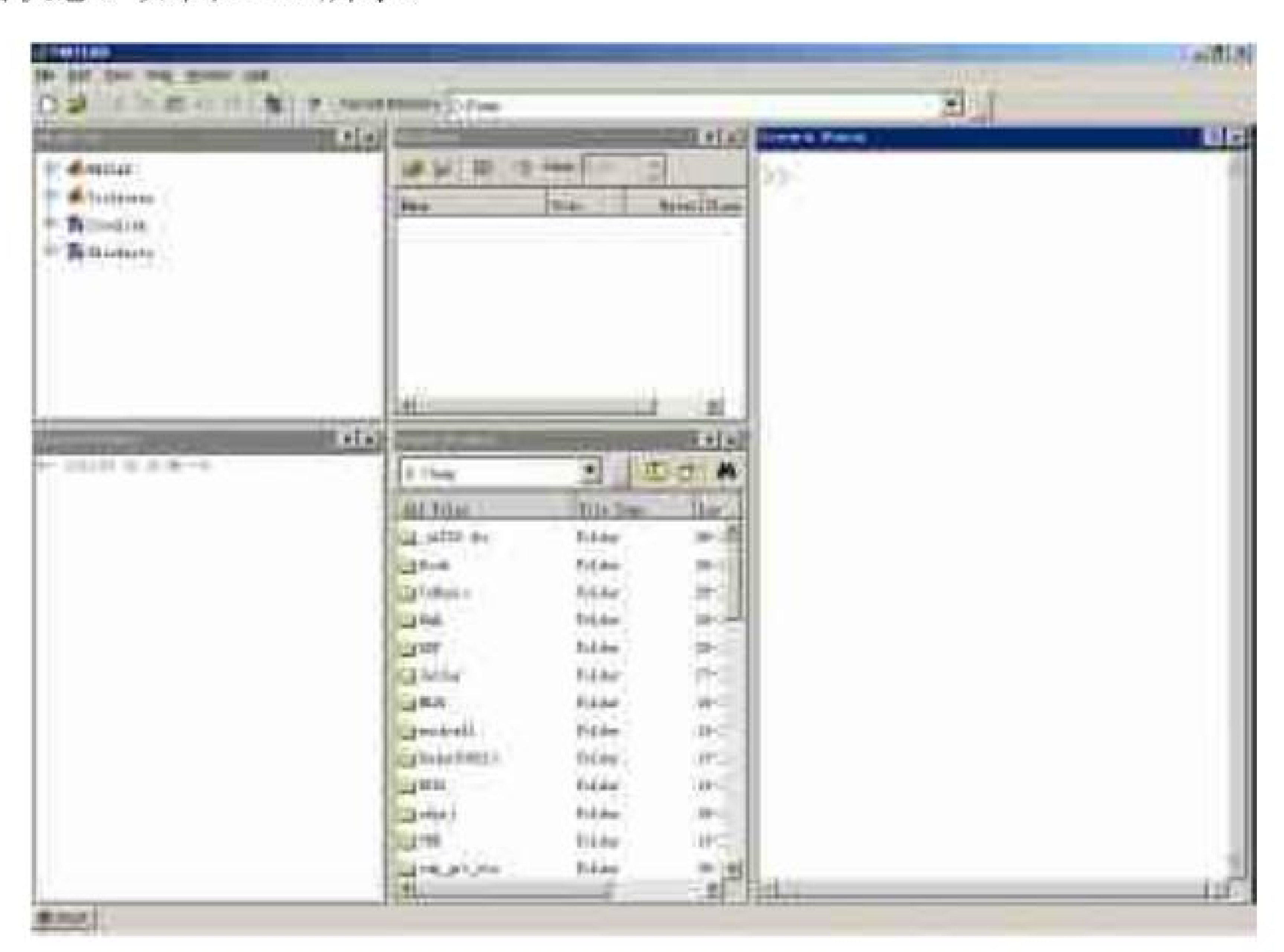


图 1-12　选择 Five Panel 菜单命令的界面

在 MATLAB 用户界面的 View 菜单下还有一些菜单命令可以用来选择显示在图形界面中的窗口，用户可以根据自己的喜好选择配置用户界面。在本小结提及的各种用户交互的窗口将在后续的章节中详细讲述。一般情况下，建议用户选择“Command Windows Only”方式来运行 MATLAB。在这种启动方式下，MATLAB 的启动速度比较快，占用的资源略少。

1.3　Command Windows 和 MATLAB 指令

MATLAB 的功能是通过大量的函数或者指令来实现的，这些函数有些可以通过 MATLAB 的图形用户界面直接使用，而大多数的函数都是通过 MATLAB 的命令行窗口，由用户直接键入相应的函数或命令来调用的。通过命令行窗口使用 MATLAB 是最基本的方法。

1.3.1　命令行窗口

MATLAB 的命令行窗口不仅可以内嵌在 MATLAB 的用户界面中，还可以浮动在界面上，单击命令行窗口上的按钮，就可以浮动命令行窗口，如图 1-13 所示。

若希望重新将命令行窗口嵌入到 MATLAB 的界面中，可以执行“View”菜单下的“Dock Command Window”命令即可。

MATLAB 的命令行窗口无论是外观还是使用方法，从其 4.x 的版本起就已经没有明显的变化了，它最具特色的就是其命令回调的功能，也就是说在 MATLAB 的命令行窗口键入任意算术表达式，系统将自动解算，并给出结果，见例子 1-1。



图 1-13　浮动的 MATLAB 命令行窗口

例子 1-1　计算算术表达式 $\frac{-5}{(4.8+5.32)^2}$ 。

只要直接在 MATLAB 的命令行窗口中键入：

```
>> - 5/(4.8+5.32)^2↙
```

系统将直接计算表达式的结果，并且给出答案：

```
ans =
        - 0.0488
```

注意：

? 这里的符号“>>”为 MATLAB 的命令行提示符。

? 这里的符号“↙”表示键入表达式之后按回车键。

? MATLAB 的数学运算符同其他的计算机高级语言(例如 C 语言)类似。

? 这里计算得到的结果显示为 ans，ans 是英文单词“answer”的缩写，它是 MATLAB 默认的系统变量。

? 所有 MATLAB 的计算结果和数值都默认使用双精度类型显示。

例子 1-2　计算复数的运算(1+2i)×(1- 3i)。

在 MATLAB 命令行窗口中键入：

```
>> (1+2i)*(1- 3i) ↙
```

系统直接计算表达式的结果，并给出答案：

```
ans =
        7.0000- 1.0000i
```

注意：

在 MATLAB 中表示复数，按照例子 1-2 中所示的样式，即 x±yi，其中 x 和 y 都是双精度的数字。在这里，i 作为复数单位存在，同样也可以使用 j 表示复数单位。

上面的两个例子中都是将 MATLAB 直接作为计算器来使用的，在 MATLAB 的命令窗口中还可以定义相应的 MATLAB 数据对象和变量以及调用函数。

例子 1-3　调用函数。

```
>> cos(pi/2)
ans =
        6.1232e-017
>> exp(acos(0.3))
ans =
        3.5470
```

在例子 1-3 中调用余弦函数求 p/2 的余弦值，但是一般的数学知识告诉我们 p/2 的余弦应该为 0，但是 MATLAB 求的数值不是 0，而是一个近似为 0 的数值，这都是由 MATLAB 浮点数的计算精度引起的。在调用函数的时候，需要注意括号的作用，它会造成主计算优先级的变化。例子 1-3 在计算第二个表达式的时候，首先计算反余弦函数，然后再计算指数函数。

MATLAB 的功能是通过大量的 M 语言函数或者 MATLAB 内建的指令来完成的，在命令行窗口中，调用这些函数的方法就是直接键入函数或者指令，并且根据不同的函数提供相应的参数列表。MATLAB 的命令行窗口具有命令行记忆的功能，也就是说，在命令行窗口中，使用上、下光标键就可以重复以前键入的指令了，这对使用 MATLAB 是非常便利的功能。而且 MATLAB 还具有局部记忆的功能，例如在 MATLAB 的命令行窗口中曾经执行了一个函数 testcommandwindows，那么再次运行该函数时，只要在命令行中键入 test，然后按光标上键?，整条命令就会出现在命令行窗口中，这时按回车键就可以执行该指令了。

1.3.2　设置命令行窗口的显示方式

其实 MATLAB 的计算结果除了用图形方式进行可视化输出以外，在大多数情况下，都是在命令行窗口中输出的，而且命令行窗口中的文本输出形式，例如文本的字体、字号或者色彩等都可以根据用户的需要自定义。设置的方法是执行“File”菜单下的“Reference”命令，在弹出的对话框中，选择左边选项中的“Command Window”项，然后展开“Font & Colors”子选项，对话框的右边则出现可以设置的文本输出特性，如图 1-14 所示。

一旦通过图 1-14 所示的对话框对各种文本的显示属性进行了设置，在单击“Apply”按钮或者“OK”按钮时属性值就会立即生效，而且设置的属性也会被永久保留下来，在下一次启动 MATLAB 时将直接使用设定的属性。

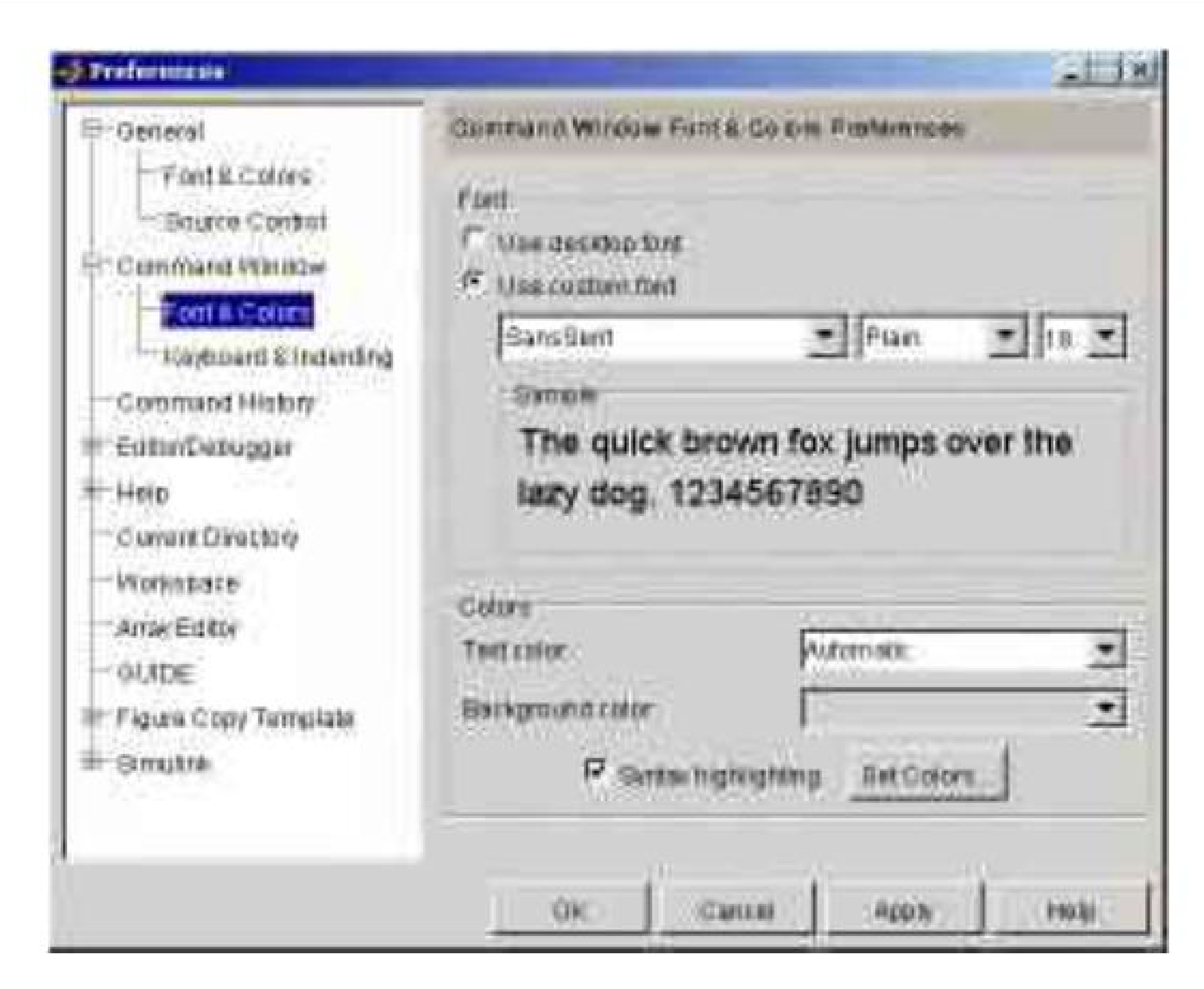


图 1-14　设置命令行窗口的文本属性

在命令行窗口中显示数值计算的结果具有一定的格式，例如在上一小节的两个例子中，所有的数值都是按照 MATLAB 默认的数字显示格式——短(short)格式显示的，在这种表示方法下具有固定的格式，保留小数点后四位有效数字，对于大于 1000 的数值，使用科学计数法表示。设置数据的显示格式需要使用 format 指令，具体的使用方法如表 1-2 所示，在表格中使用的示例数据为自然对数的底数。

表 **1-2**　**MATLAB** 命令行窗口显示数据的格式

指　令	说　　明	示　例
format	默认的数据格式，同 short 格式一致	271.82 显示为 271.8200 2718.2 显示为 2.7182e+003
format short	具有固定的显示格式，保留小数点后四位有效数字，对于大于 1000 的数值，使用科学计数法表示	
format long	具有固定的 15 位有效数字	2.71828182845905
format short e	具有 5 位有效数字的科学计数法表示	2.7183e+000
format long e	具有 15 位有效数字的科学计数法表示	2.718281828459046e+000
format short g	紧凑的显示方法，在 format short 和 format short e 中自动选择数据的显示格式	2.7183
format long g	紧凑的显示方法，在 format long 和 format long e 中自动选择数据的显示格式	2.71828182845905
format hex	使用十六进制的数据形式表示	4005bf0a8b14576a
format +	在使用该格式显示大矩阵时，分别使用正号、负号或者空格显示矩阵元素中的正数、负数或者 0	+
format bank	使用金融的数据显示方法，小数点后只有两位有效数字	2.72
format rat	使用近似的分数表示数值	1457/536

注意：

利用表格 1-2 中的指令设定的数据格式仅在当前的 MATLAB 会话的命令行窗口中有效。

例子 1-4　使用不同的数据显示格式显示数字。

在 MATLAB 命令行中，键入下面的指令：

```
>> pi
ans =
       3.1416
>> format long
>> pi
ans =
       3.14159265358979
>> format +
>> pi
ans =
+
```

例子 1-4 中使用 MATLAB 的内建函数 pi 获取常数 p 的数值，也可以将 pi 看作是 MATLAB 的常数。MATLAB 的常数将在后面的章节中详细讲述。从例子 1-4 中可以看出，在不同的数据显示格式下，显示的数据位数不尽相同，大家可以根据自己的需要，设置数据显示的位数。

1.3.3　常用的控制指令

MATLAB 包含的函数可以粗略地分为两大类，其中之一是执行各种具体计算或者数据处理功能的函数，例如 cos 函数、sqrt 函数等，而另外一类是进行用户环境控制的指令，比如退出 MATLAB 会话、执行操作系统的功能等。在表 1-3 中对一些常用的控制指令进行了总结。

表 **1-3**　常用的 **MATLAB** 控制指令

指　令	说　　明
exit、quit	退出 MATLAB 会话
format	数字格式
clc	清除当前的命令行窗口
home	将当前命令行窗口的光标设置在左上角
dos	执行 dos 系统指令
unix	执行 unix 系统指令
system	执行系统指令，针对不同的系统有不同的指令
perl	执行 perl 脚本
cd	切换路径或者显示当前的路径
pwd	显示当前的路径
dir、ls	显示当前路径下的文件
what	显示当前路径下的 MATLAB 文件
which	判断当前文件的所在路径

熟悉不同操作系统应用的用户可以看到，MATLAB 的控制指令能够兼容不同的操作系统——Windows 平台和 Unix 平台。

例子 1-5　常用的控制指令示例。

```
>> %察看当前的路径
>> pwd
ans =
D:\Temp
>> %显示当前路径下的文件和子目录
>> dir
.
..
FigureMenuBar.fig
How am I suppose to live without you.mp3
Java
Said I loved you ,but I lied.mp3
eagles_hotelcanifornia.mp3
gui_soln.fig
gui_soln.m
mymesh.m
test.m
>> %显示当前路径下的 MATLAB 文件
>> what
M-files in the current directory D:\Temp
gui_soln      mymesh       test
>> %what 指令的路径
>> which what
what is a built-in function.
>> %察看 M 文件的路径
>> which logo
E:\MATLAB6p5\toolbox\matlab\demos\logo.m
>> %执行系统指令
>> system('copy eagles_hotelcanifornia.mp3 eagles.mp3')
已复制              1 个文件。
ans =

     0
```

依次执行例子 1-5 的指令可以得到相应的指令输出，注意 what 指令和 dir 指令的区别。另外，在执行系统指令的时候还可以使用 MATLAB 的“!”符号，例如在执行例子 1-5 最后的拷贝命令时，也可以这样做：

```
>> !copy eagles_hotelcanifornia.mp3 eagles.mp3
```

两者的区别在于通过 system 指令执行系统命令能够获取系统指令的返回值，例如指令执行的状态等。system 指令的一般使用方法为

```
system('command');
```

其中，command 就是系统指令，用单引号“'”括起来作为参数传递给 system 指令。

在使用 which 指令的时候得到的输出根据 which 指令后面的参数不同而不同，例如在执行 which what 时，系统判断 what 为内建(build-in)的函数，而在执行 which logo 指令时，系统判断 logo 为 M 文件，并且给出了 M 文件所在的路径。有关内建函数或者 M 文件函数的概念将在本书的第四章中详细讲述。

1.4　Command History 和历史记录

MATLAB 的命令行窗口提供了非常友好的交互能力，用户可以在这个环境中边思考边验证，可以随时针对自己的想法进行验证与考察，在用户完成了设计之后，就可以将那些已经经过验证考察的指令通过 MATLAB 的记录命令能力，将这些指令再次提取出来。这种记录指令的能力就是利用 MATLAB 提供的 Command History 窗口以及相应的 MATLAB 控制指令完成的。

1.4.1　命令行历史窗口

在默认的 MATLAB 界面中，命令行历史窗口总是在 MATLAB 界面的左下角，和命令行窗口类似，命令行历史窗口也可以浮动出来，单击命令行历史窗口界面上↗按钮，就可以浮动该窗口，如图 1-15 所示。同样，通过“View”菜单下的“Dock Command History”指令也可以将命令行历史窗口内嵌回 MATLAB 的界面中。



图 1-15　MATLAB 的历史记录窗口

在命令行历史窗口中主要记录了在 MATLAB 命令行窗口中键入的所有指令，一般包括每次启动 MATLAB 的时间，以及每次启动 MATLAB 之后键入的所有 MATLAB 指令。这些指令不但可以清楚地记录在命令行历史窗口中，而且还可以被再次执行，它们不仅能够被复制到 MATLAB 的命令行窗口中，而且还可以通过这些指令的记录直接创建 M 文件，这些功能都可以通过命令行历史窗口的快捷菜单来方便地完成，如图 1-16 所示。



图 1-16　命令行历史的快捷菜单

快捷菜单中的指令说明如下：

- Copy：拷贝当前选中的指令，可以将指令粘贴到其他的应用程序窗口中。
- Evaluate Selection：执行当前选中的指令。
- Create M-File：把当前选中的指令创建一个新的 M 文件，文件的内容就是选中的所有指令。
- Delete Selection：从命令行历史窗口中删除当前选中的指令。
- Delete to Selection：将当前选中指令之前的所有历史记录指令从命令行历史窗口中删除。
- Delete Entire History：删除命令行历史窗口中所有的指令。

例子 1-6　命令行历史窗口的应用。

继续前面例子 1-5 的应用，在命令行历史窗口中，输入下面几条命令(如图 1-17 所示)：

```
%执行系统指令
system('copy eagles_hotelcanifornia.mp3 eagles.mp3')
clear all
```

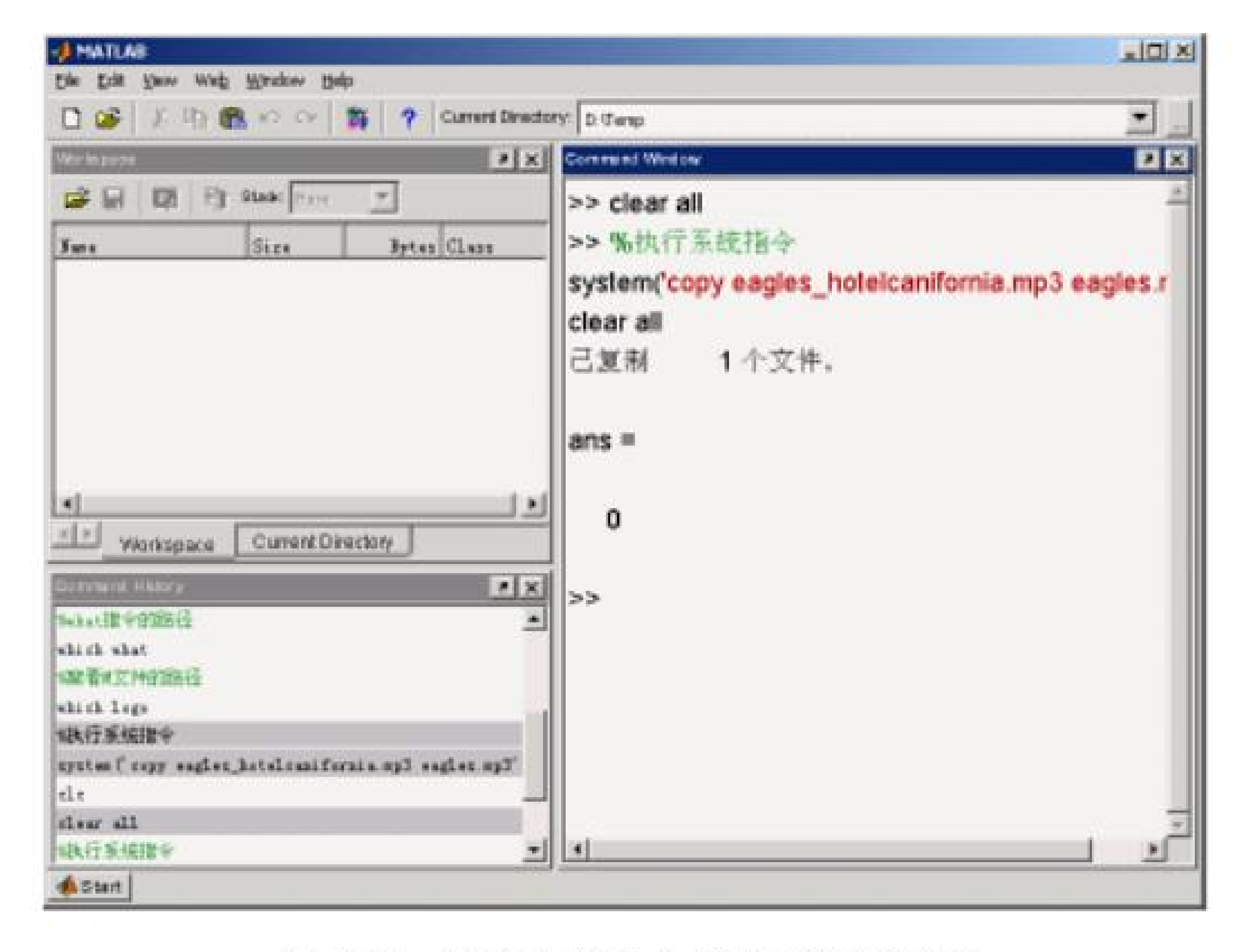


图 1-17　选择多条命令行历史语句运行

注意：

选择多条指令时，可以按住 Ctrl 键或者 Shift 键来达到选中多条语句的目的。如果使用 Ctrl+A 键，可以选中所有命令行历史窗口中的指令。

然后单击鼠标右键，在弹出的快捷菜单中，选择“Evaluate Selection”命令，重复运行这些指令。

执行单条指令的方法更简便，只要在命令行窗口中用鼠标左键双击指令就可以了。

MATLAB 主要将所有历史命令都保存在一个历史记录文件中，这个文件位于系统路径下，一般不需要进行编辑。用户可以通过设置命令行历史窗口的属性来设置有关文件，执行“File”菜单下的“References”命令，在命令行历史窗口属性设置对话框中，可以设置有关命令行历史窗口的属性，如图 1-18 所示。



图 1-18　命令行历史窗口的属性设置

有关命令行历史窗口属性参数的具体意义，请参阅 MATLAB 的帮助文档。

1.4.2　diary 指令

diary 指令也是常用的 MATLAB 命令行指令之一，该指令的功能是创建一个日志文件，在这个文件中，能够把所有在 MATLAB 命令行键入的指令以及在命令行窗口的输入保存下来。这个日志文件为纯文本格式，可以利用任何一种文本编辑器编辑这个文件。

diary 指令的常用方法如下：

- diary：在历史日志记录指令 On 和 Off 状态之间切换。
- diary on：打开历史日志记录功能。
- diary off：关闭历史日志记录功能。
- diary('filename')：创建日志文件，文件名为 filename。

在使用 diary 指令时，若不指定文件名，则 MATLAB 自动创建一个默认文件名 diary(注意，该文件没有扩展名，为纯文本文件)的日志文件，并且进入到历史日志记录状态。在日志文件中将记录所有在命令行窗口中键入的指令以及这些指令运行的结果。注意，diary 记录指令的功能也仅在执行 diary 指令之后的 MATLAB 会话中有效，一旦关闭了 MATLAB 再次启动时，则 diary 指令需要重新键入。

1.5　Current Directory 和搜索路径

单纯的 MATLAB 核心模块就包含了 400 余个核心函数和数百个 M 文件函数，如果再包含了其他工具箱的函数，则 MATLAB 的函数数量就有成千甚至上万个，另外再加上用户自己开发的算法文件，整个 MATLAB 就是由若干 MATLAB 文件、数据构成的庞大的软件体系。那么，在用户执行了一条指令或者加载了一个数据文件时，MATLAB 是如何判断这些文件所处的位置，并按照要求加载正确的文件的呢？这一切都是利用了 MATLAB 的路径管理方法——搜索路径完成的。

1.5.1　Current Directory 当前路径察看器

MATLAB 加载任何文件、执行任何指令都是从当前的工作路径下开始的，所以 MATLAB 也提供了当前路径的浏览器——Current Directory，该工具在默认的情况下位于 MATLAB 界面的左上方，在工作空间浏览器的下面，可以单击“Current Directory”标签切换界面。和其他的桌面工具类似，当前路径浏览器不仅可以浮动在所有窗口上方，而且还可以像默认的状态那样内嵌在桌面工具中，浮动的窗口如图 1-19 所示。

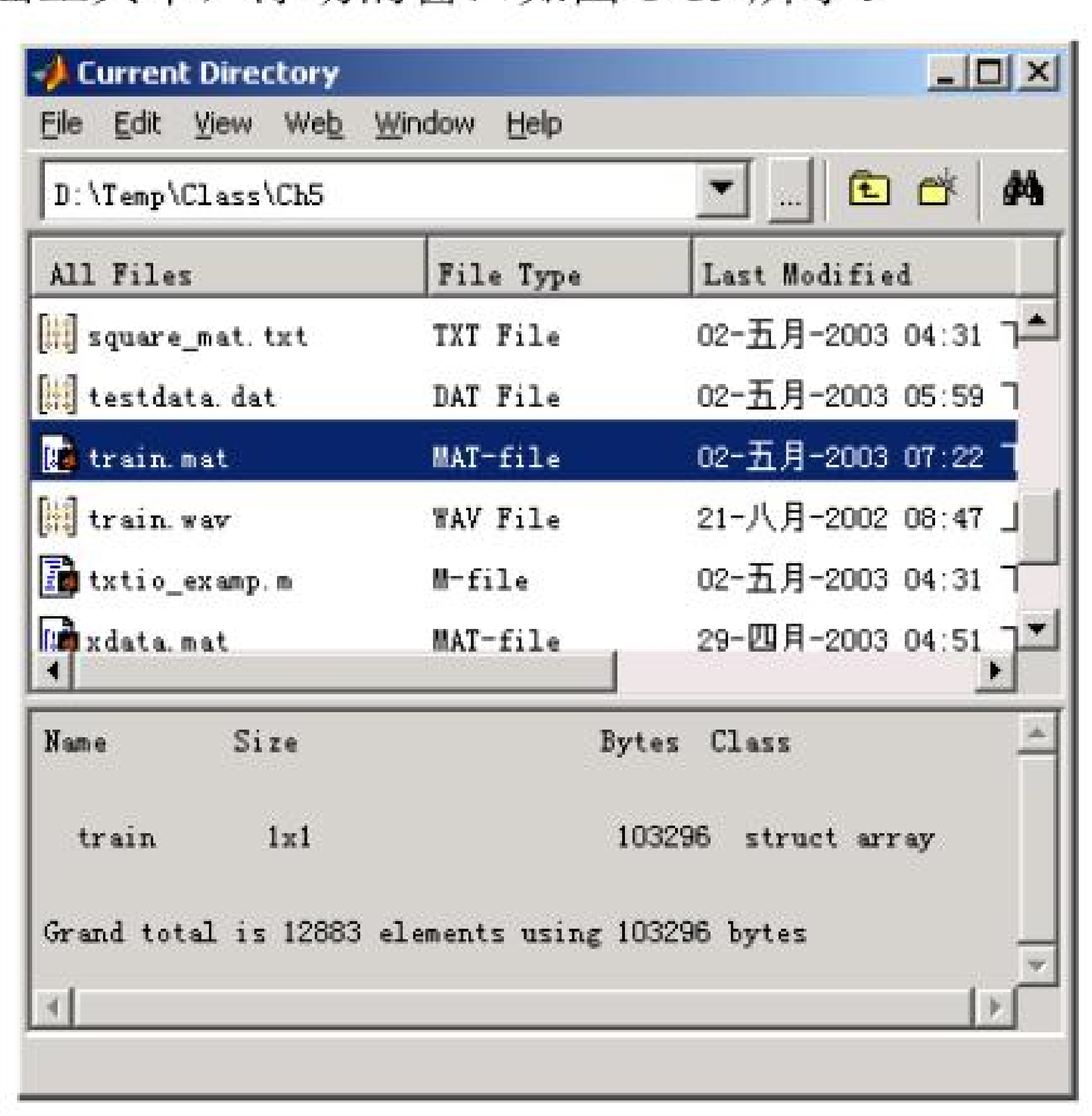


图 1-19　浮动的当前路径察看器

在如图 1-19 所示的完整的路径察看器中，还可以察看必要的 M 文件和 MAT 文件信息，如果在 M 文件中编写了帮助文档，则相应的帮助信息将显示在窗口的下方，同样 MAT 文件中包含的变量信息也会显示在这里。

当前路径察看器的主要作用是帮助用户组织管理当前路径下的 M 文件，并且通过该工具，能够运行、编辑相应的文件，加载 MAT 数据文件等，这些操作都可以通过对应的右键快捷菜单完成。当前路径察看器的快捷菜单命令虽多，但是功能一目了然，这里就不再赘述了，请大家察看相应的帮助文档，或者直接使用菜单命令来察看运行的效果。

当前路径察看器也可以通过相应的属性设置对话框设置其显示的外观。执行“File”菜单下的“Preferences”命令，弹出如图 1-20 所示的对话框。

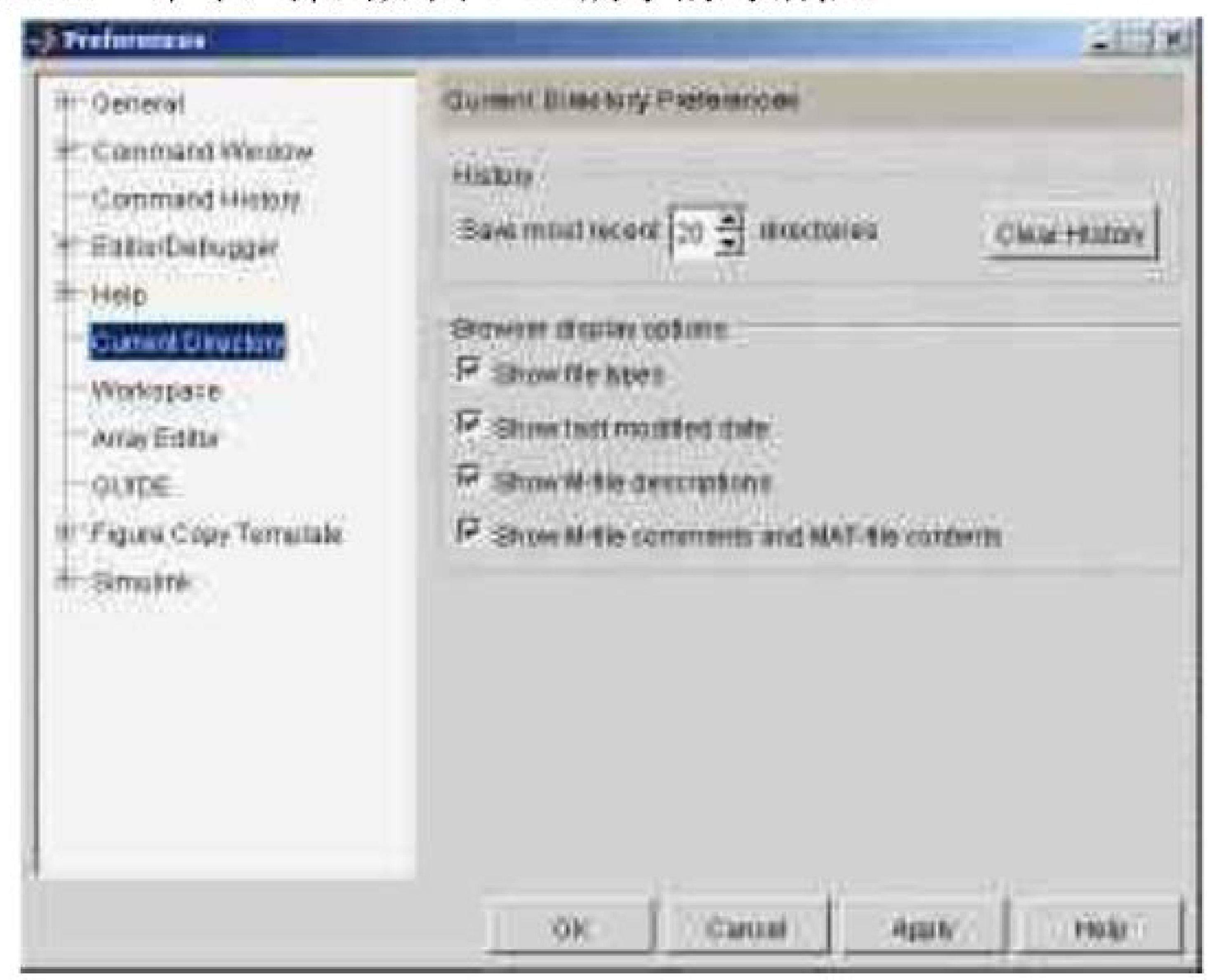


图 1-20　设置当前路径察看器的属性

在对话框中可以设置有关当前路径察看器的属性，其中比较重要的就是 History 栏目，这里主要设置在 MATLAB 界面中 Current Directory 下拉框中显示的路径个数，默认的数值为 20 个。Current Directory 下拉框能够将最近访问过的 20 个路径信息保存起来，便于用户快速地回访到那些已经访问过的路径。有些时候发生了路径的变化，或者不需要保留这些路径信息时，则可以单击“Clear History”按钮，将该下拉框中的路径信息删除，删除后，仅保留当前的工作路径。

1.5.2　工作路径

MATLAB 的工作路径就是当前的路径，MATLAB 启动时采用默认的路径作为当前工作路径，这样所有的 MATLAB 文件保存和读取都将从该路径下开始。在 Windows 平台下，MATLAB 提供了一个路径：$matlabroot\work，在 MATLAB 首次启动时，就会使用该路径作为当前的工作路径。

注意：

对于 Unix 平台，将当前启动 MATLAB 的路径作为工作路径。

启动时选择的工作路径是可以修改的，一般来说，可以利用 MATLAB 的 Startup 脚本文件完成，该脚本文件在 MATLAB 启动时自动被执行，可以将设置工作路径的指令(cd 指令)添加到该文件中，则每次启动 MATLAB 的时候将自动地切换到该路径下。

对于 Windows 平台还有一种方法进行路径设置。在桌面上选择 MATLAB 应用程序的快捷图标，然后单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，在弹出的对话框中有一个“起始位置”文本输入框，可以将需要设置的启动工作路径设置在这里，如图 1-21 所示。



图 1-21　设置工作起始位置

需要注意的是，对于同一个应用程序的不同的快捷方式可以设置不同的起始位置。

注意：

尽管 MATLAB 提供了一个工作路径$matlabroot\work，但是该路径并不适合保存所有的工作文件，用户应该尽量为自己创建一个工作路径，将必要的文件保存在自己的工作路径下。而且出于运行可靠和方便的目的，需要将自己的工作路径设置为当前的路径。若需要切换工作路径，则可以单击“Current Directory”下拉框边上的...按钮，通过路径浏览对话框设置不同的工作路径。

1.5.3　搜索路径

如前文所述，MATLAB 的文件是通过不同的路径来进行组织管理的，为了避免执行不同路径下的 MATLAB 文件而不断切换不同的路径，MATLAB 提供了搜索路径机制来完成对文件的组织和管理。

所有的 MATLAB 文件都被保存在不同的路径中，那么将这些路径按照一定的次序组织起来，就构成了搜索路径。当执行某个 MATLAB 指令时，系统将按照以下的顺序搜索该指令：

- 首先判断该指令是否为变量。
- 然后判断该指令是否为内建的函数。
- 接着在当前的路径下搜索是否存在该指令文件。
- 最后从搜索路径中依次搜索该文件直到找到第一个符合要求的 M 文件为止。
- 若上述的搜索都没有找到该指令，则报告错误信息。

MATLAB 按照上面的顺序来判断指令的执行，并且仅执行第一个符合条件的指令。

注意：

实际的指令解析顺序要更复杂一些，将在本书后面的章节中再次详细讲述。

设置搜索路径可以通过 MATLAB 指令，也可以通过对话框界面完成。执行“File”菜单下的“Set Path”指令，在弹出的对话框中可以设置相应的搜索路径，如图 1-22 所示。

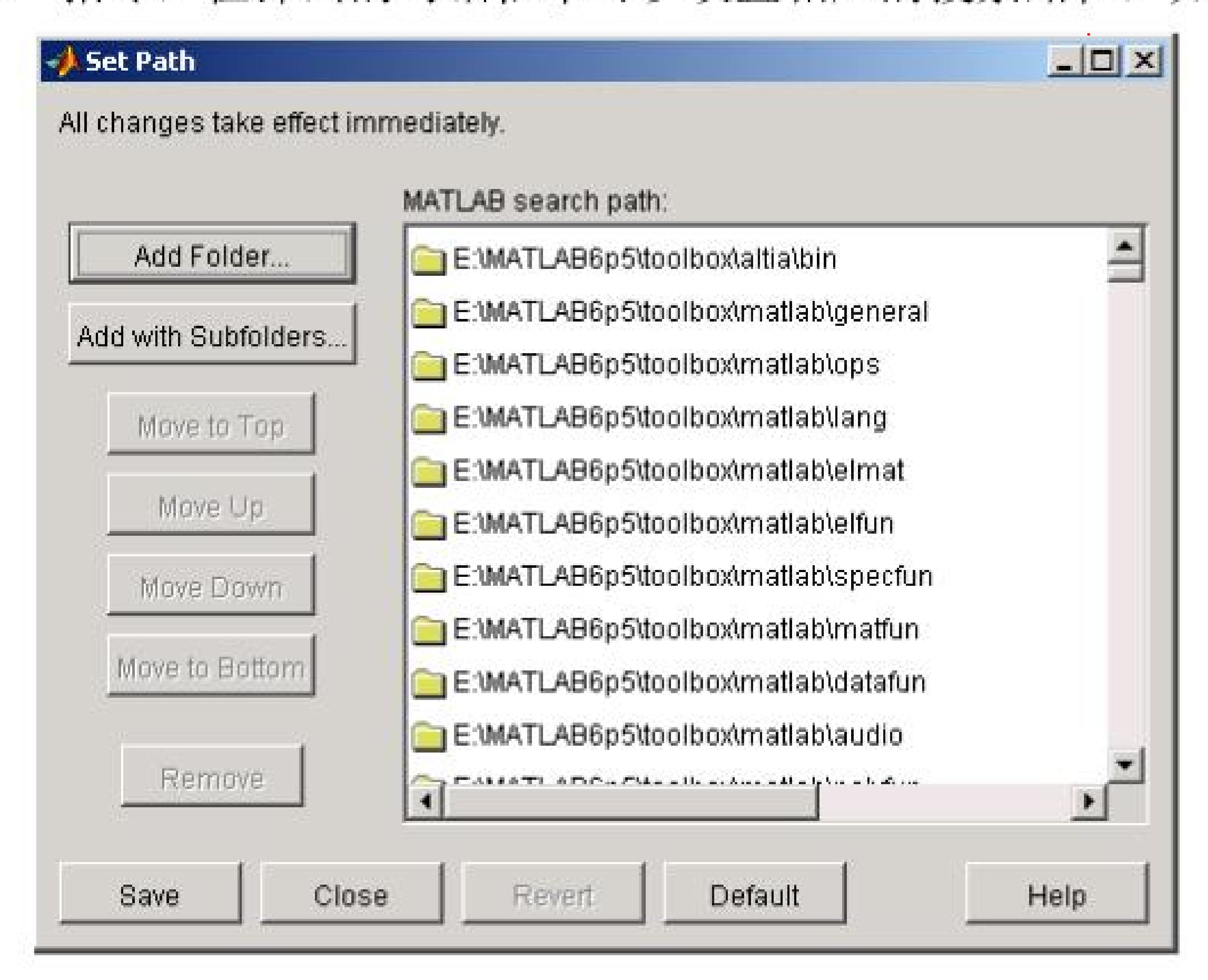


图 1-22　MATLAB 的搜索路径设置对话框

通过“Add Folder”或者“Add with Subfolders”按钮将路径添加到搜索路径列表中，对于已经添加到搜索路径列表中的路径可以通过“Move to Top”等按钮修改该路径在搜索路径中的顺序，对于那些不需要出现在搜索路径中的内容，可以通过“Remove”按钮将其从搜索路径列表中删除。

在修改完搜索路径之后，则需要保存搜索路径，这时单击对话框中的“Save”按钮就可以完成该工作。单击“Save”按钮时，系统将所有搜索路径的信息保存在一个 M 文件中——pathdef.m，有兴趣的读者可以察看该文件的内容，通过修改该文件也可以修改搜索路径。有关搜索路径的详细信息请参阅 MATLAB 的帮助文档。

以上设置路径的方法也可以通过指令来完成，这些指令如下：

- path：察看或者修改路径信息。
- addpath：添加路径到搜索路径中。
- rmpath：将路径从搜索路径列表中删除。
- path2rc：保存搜索路径信息。
- pathtool：显示搜索路径设置对话框。
- genpath：生成路径字符串。

关于这些指令的使用，见例子 1-7。

例子 1-7　设置 MATLAB 搜索路径的指令使用示例。

```
>> %显示当前的搜索路径信息
>> path
        MATLABPATH
```

```
    E:\MATLAB6p5\toolbox\matlab\general
    E:\MATLAB6p5\toolbox\matlab\ops
    E:\MATLAB6p5\toolbox\matlab\lang
    E:\MATLAB6p5\toolbox\matlab\elmat
    ⋮
>> %生成路径字符串
>> p = genpath(pwd)
p =
D:\TEMP;D:\TEMP\Class
>> %添加搜索路径
>> addpath(p,'-end')
>> %察看路径信息
>> path
        MATLABPATH
    E:\MATLAB6p5\toolbox\matlab\general
    E:\MATLAB6p5\toolbox\matlab\ops
    E:\MATLAB6p5\toolbox\matlab\lang
    E:\MATLAB6p5\toolbox\matlab\elmat
    ⋮
    D:\TEMP
    D:\TEMP\Class
```

在例子 1-7 中主要使用了 genpath 命令从当前的路径中生成路径字符串，使用 addpath 命令将路径字符串添加到搜索路径的末端。有关这些函数(指令)的详细说明请参阅MATLAB 的帮助文档。

在 MATLAB Release 13 中为了提高系统的运行性能，提供了一个新特性——Toolbox Path Cache，该特性将所有 MATLAB 工具箱路径和路径下面的文件名称保存在 Cache 文件中，这样，在调用工具函数的时候，就能够大大提高程序调用的速度。在每次启动 MATLAB 的时候都可以看到如下的信息：

```
Using Toolbox Path Cache. Type "help toolbox_path_cache" for more info.
```

在使用新版的 MATLAB 时，不要将用户自己定义的 MATLAB 文件随意地添加到工具箱路径下，也不要任意地修改工具箱路径下已有的文件。因为修改后的文件很有可能没有被重新加载到 Toolbox Path Cache 中，而且一旦重新安装了 MATLAB 或者卸载了 MATLAB 则已有的工作不会被保留下来。用户可以通过属性设置对话框，设置有关工具箱路径高速缓存的属性，若用户不需要使用高速缓存的时候，则取消对复选框“Enable toolbox path cache”的选择。

另外在 MATLAB 安装完毕首次运行的时候，还会出现如下信息：

```
MATLAB Toolbox Path Cache is out of date and is not being used.
```

用户不必理会此信息，在下一次启动 MATLAB 的时候，更新后的高速缓存将直接发挥作用。

一般来说，在对 MATLAB 工作路径的文件进行了修改之后，需要更新工具箱路径高速缓存，或者在针对 MATLAB 的部分模块进行了更新升级之后，也需要更新工具箱路径高速缓存。其实每次 MATLAB 在启动的时候，都会检查路径缓存，并且进行必要的更新。在需要人工干预的时候，可以单击属性设置对话框上的“Update Toolbox Path Cache”按钮，或者使用指令 rehash。设置路径高速缓存的界面如图 1-23 所示。

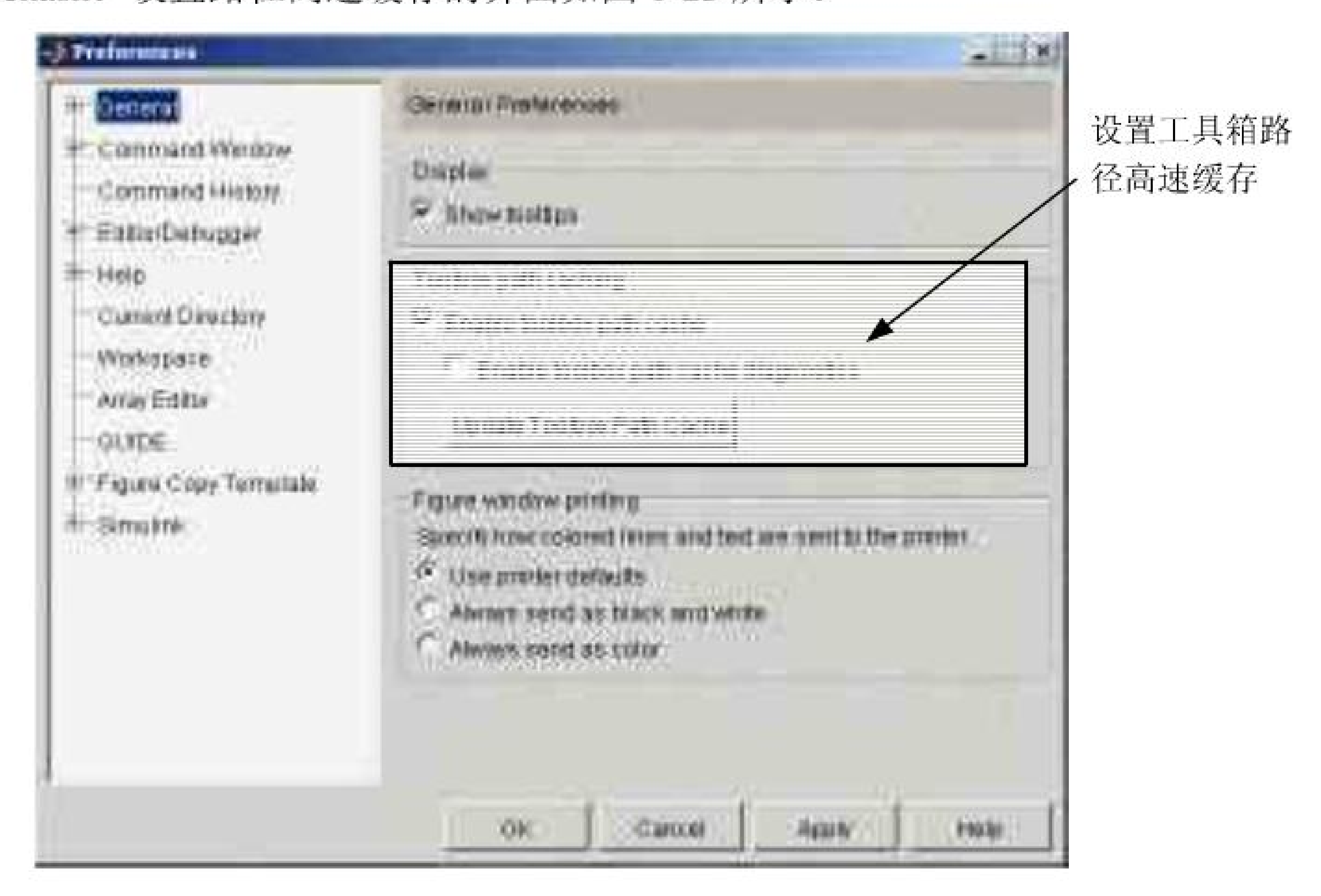


图 1-23　设置路径高速缓存

若需要 MATLAB 统计启动时间的时候，可以选择“Enable toolbox path cache diagnostics”复选框，这样每次启动 MATLAB 的时候，将显示启动消耗的时间。

1.6　Launch Pad 和 Start 菜单

MATLAB 的 Launch Pad 和 Start 菜单从功能上看非常相似，所以在新版本的 MATLAB 6.5 中，就没有将 Launch Pad 作为默认的 MATLAB 界面工具。无论是通过 Launch Pad 还是 Start 菜单，都能够访问、使用所有 MATLAB 产品的资源，包括文档、工具、演示示例等。

Launch Pad 提供了一个简单的窗口界面，具有 MATLAB 产品的树状列表，通过树状列表就可以访问所有的资源。在 Start 菜单界面上以菜单的形式显示所有工具的列表，大家可以像访问 Windows 的“开始”菜单一样来使用 MATLAB 的 Start 菜单。

悬浮的 Launch Pad 窗口如图 1-24 所示。这里将所有的 MATLAB 产品分为四类：MATLAB、Toolboxes、Simulink 和 Blocksets，单击任何一类名称前的加号，则可以显示该类产品下的产品、工具等，例如在图 1-25 中，显示了 MTALAB 产品下的工具以及相关的工具箱。

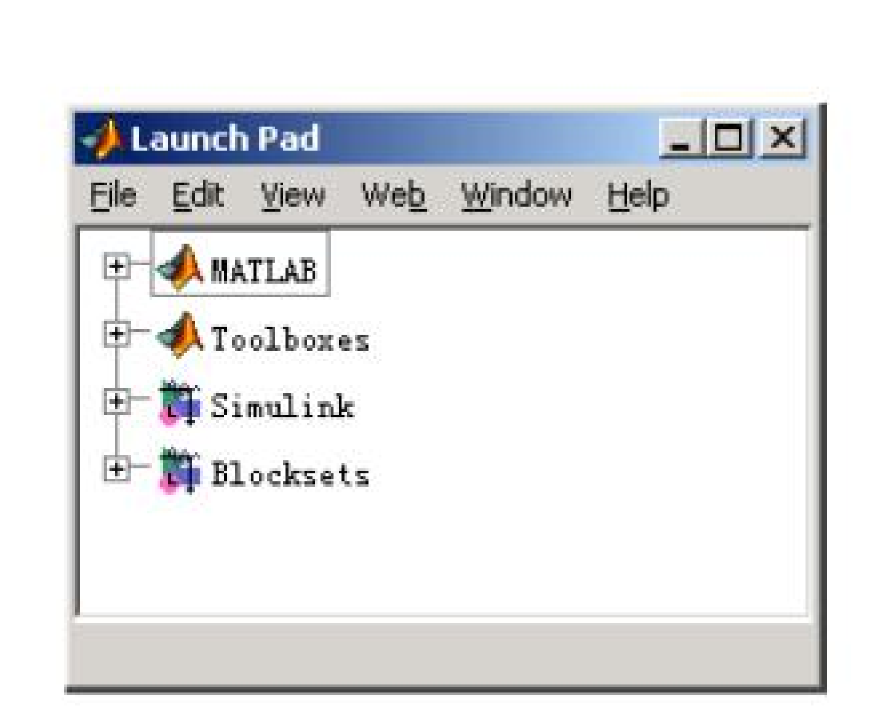


图 1-24　Launch Pad 窗口



图 1-25　展开工具的 Launch Pad 窗口

和 Launch Pad 类似，MATLAB 的 Start 菜单也包含相应的分组内容，如图 1-26 所示。

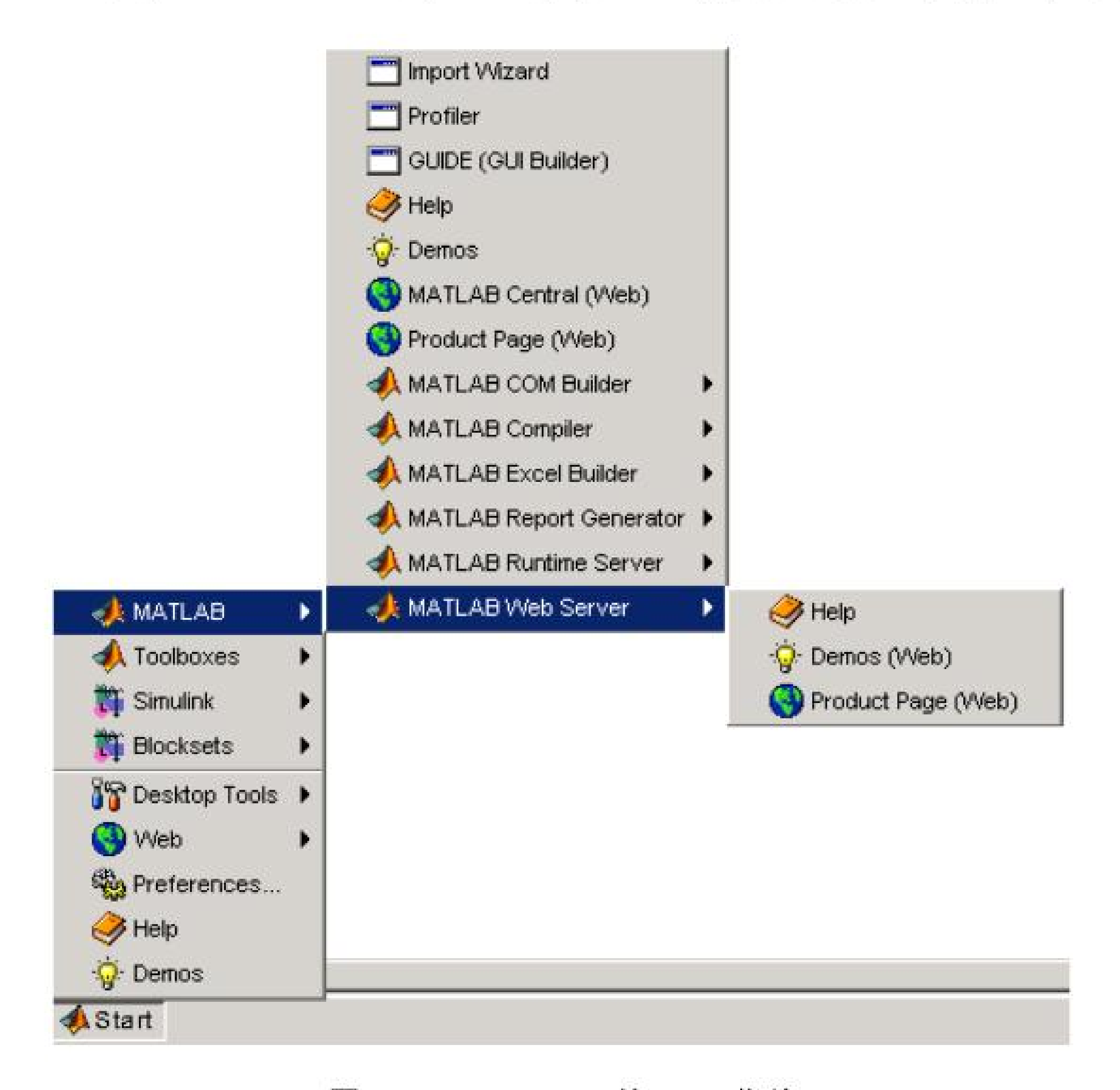


图 1-26　MATLAB 的 Start 菜单

在 Launch Pad 和 Start 菜单上主要有四类图标，它们的意义分别如下：

：可用工具，例如 MATLAB 中的 GUIDE。

：MATLAB 的帮助文档。

：MATLAB 系统自带的演示示例。

：MATLAB 的网上资源，包括产品说明等。

大家可以根据自己的使用爱好，选择 Launch Pad 或者使用 Start 菜单完成相应的功能。

1.7　使 用 帮 助

对于任何 MATLAB 的使用者，都必须学会使用 MATLAB 的帮助系统，因为 MATLAB 和相应的工具箱包含了上万个不同的指令，每个指令函数都对应着一种不同的操作或者算法，没有哪个人能够将这些指令都清楚地记忆在脑海中，而且 MATLAB 的帮助系统是针对 MATLAB 应用的最好的教科书，讲解清晰、透彻，所以养成良好地使用 MATLAB 帮助系统的习惯，对于使用 MATLAB 的用户是非常必要的。

在 MATLAB 中具有不同类型的帮助系统：在线帮助和窗口帮助。

1.7.1　在线帮助

所有的 MATALB 函数都具有自己的帮助信息，这些帮助信息都保存在相应的函数文件的注释区中，这些帮助信息是由那些编写函数的工程人员在编写函数的同时添加在函数内的，所以，这些信息能够最直接地说明函数的用途，或者函数需要的一些特殊的输入参数，以及函数的返回变量等。甚至在有些函数中，将函数采用的算法也在这里加以了说明。另外，在线帮助的获取需要通过具体的指令，才能将在线帮助显示在命令行窗口中，所以获取在线帮助的过程也非常快捷，因此，使用 MATLAB 的用户最常用的帮助就是在线帮助。获取在线帮助的方法是使用指令 help 或者 helpwin。

例子 1-8　获取在线帮助。

在 MATLAB 命令行窗口中，键入如下的指令：

```
>> %获取帮助主题
>> help
HELP topics:
matlab\general        -  General purpose commands.
matlab\ops            -  Operators and special characters.
matlab\lang           -  Programming language constructs.
matlab\elmat          -  Elementary matrices and matrix manipulation.
matlab\elfun          -  Elementary math functions.
⋮
>> %获取帮助主题下的函数列表
>> help elfun
  Elementary math functions.
  Trigonometric.
    sin          - Sine.
    sinh         - Hyperbolic sine.
    asin         - Inverse sine.
    asinh        - Inverse hyperbolic sine.
⋮
```

```
>> %获取具体函数的帮助
>> help sin
 SIN     Sine.
    SIN(X) is the sine of the elements of X.
 Overloaded methods
    help sym/sin.m
```

在例子 1-8 中，使用的省略符号是为了缩减篇幅而用，在实际的 MATLAB 中，将给出全部内容。

在线帮助不仅可以显示在命令行窗口中，还可以显示在 MATLAB 的帮助窗口中，内容仍然是在线帮助的内容，例如：

```
>> %在窗口中显示在线帮助信息
>> helpwin sin
```

这时 sin 函数的在线帮助信息将显示在帮助窗口中，如图 1-27 所示。



图 1-27　将在线帮助的内容显示在窗口中

所有的 MATLAB 函数还具有一类在线帮助，叫做 H1 帮助行，这部分内容为每一个 M 语言函数文件的在线帮助的第一行，它能够被 lookfor 函数搜索、查询，因此在这一行帮助中，往往是言简意赅的说明性语言，在所有的帮助中相对最重要。例如，在 MATLAB 命令行窗口中键入：

```
>> %使用 H1 帮助行
>> lookfor Fourier
FFT Discrete Fourier transform.
FFT2 Two-dimensional discrete Fourier Transform.
FFTN N-dimensional discrete Fourier Transform.
IFFT Inverse discrete Fourier transform.
```

IFFT2 Two-dimensional inverse discrete Fourier transform.

IFFTN N-dimensional inverse discrete Fourier transform.

⋮

这时 MATLAB 将所有有关傅立叶变换的函数罗列在命令行窗口中，这些函数的 H1 帮助行都有关键字 Fourier。

关于如何编写 MATLAB 的在线帮助和 H1 帮助行，将在本书的第四章中详细讲述。

1.7.2　窗口帮助

尽管在线帮助使用起来简便、快捷，但是在线帮助能够提供的信息毕竟有限，而且并不是所有与函数有关的内容都可以用在线帮助的形式表示，比如数学公式，图形等。因此，MALTLAB 还提供了内容更加丰富的帮助文档，作为 MATLAB 的用户指南出现。目前 MATALB 的帮助文档有英文版和日文版，而在中国地区使用的 MATLAB 只有英文版的帮助文档。

MATLAB 的帮助文档显示在 MATLAB 的帮助窗口中，单击 MATLAB 用户界面上的?按钮，将打开 MATLAB 的帮助文档界面，如图 1-28 所示。



图 1-28　MATLAB 的帮助文档界面

这里能够看到的 MATLAB 帮助文档是跟随 MATLAB 产品一同发布的文档光盘经过安装之后的超文本内容。界面中的“Contents”标签页罗列了所有产品帮助文档的目录，单击这些目录以及目录下面的文章标题，就可以在右边的窗体中具体浏览帮助信息。除此之外，还具有下面几个标签页：

- Index 标签页：关键字索引查询。
- Search 标签页：关键字全文搜索。
- Demos 标签页：MATLAB 演示例子。

■ Favorites 标签页：个人喜好的书签页。

在这些标签页中，用户使用频率最高的就是“Contents”页。一般地，学习 MATLAB 不可避免地需要阅读帮助文档，而就笔者的经验而言，直接阅读帮助文档是最直接、最有效地学习 MATLAB 的方法。

此外，使用频率最高的就是“Demos”页了，如图 1-29 所示。MATLAB 为每一个工具箱或者模块都设计了很多演示示例，通过这些例子学习 MATLAB 往往能够起到事半功倍的效果，这些演示程序的作用非常独特，往往连帮助文档都无法替代其功用。所以，对于初学者来说，在阅读帮助文档的基础上，多研习 MATLAB 的 Demos，是一种学习 MATLAB 的最佳方法。

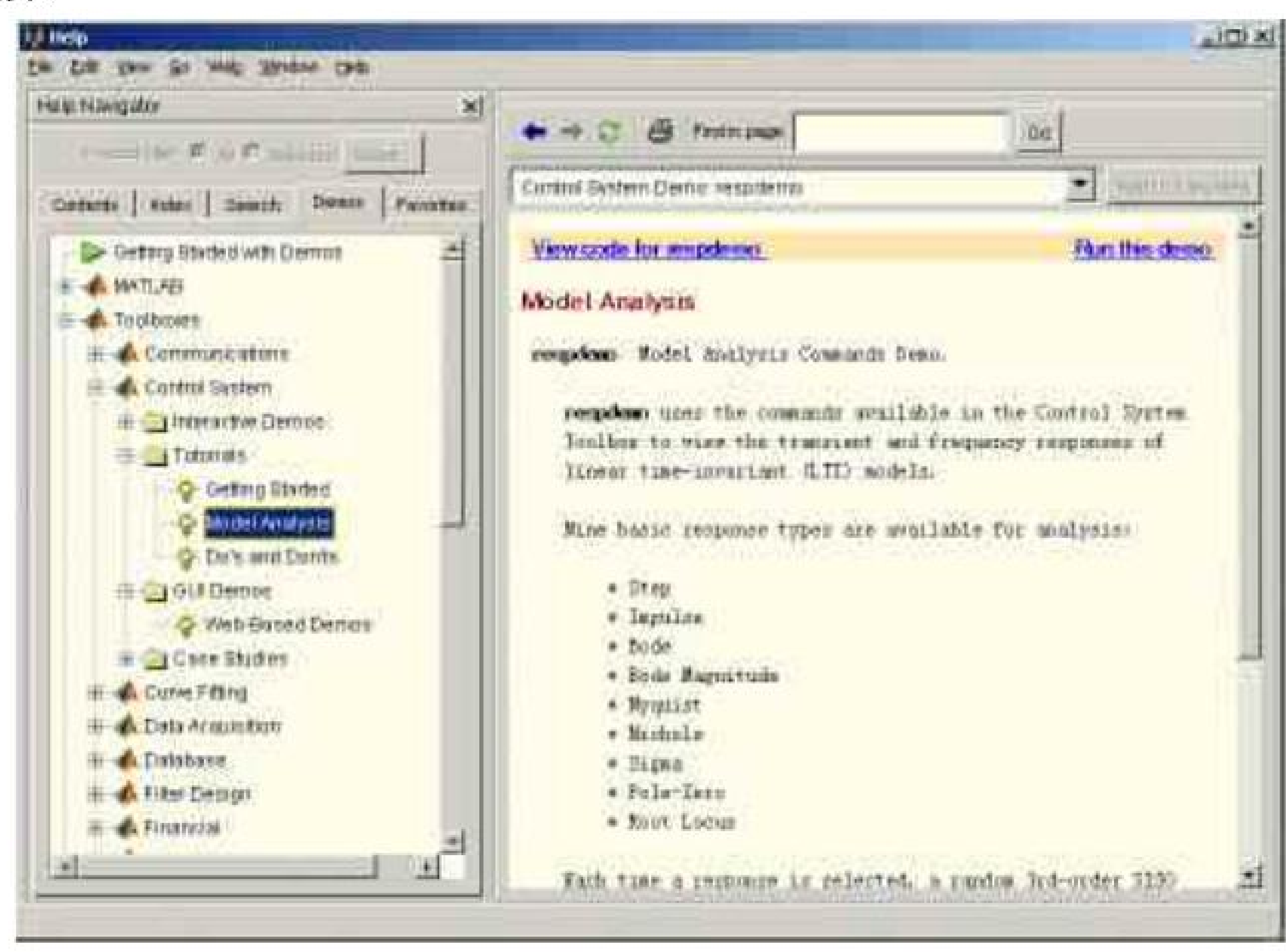


图 1-29　MATLAB 帮助浏览器的 Demos 页

MATLAB 的帮助文档除了超文本格式的以外，还具有 PDF 格式的帮助文档，这些帮助文档与 MATLAB 的产品手册(纸版)一一对应，甚至在新版的 MATLAB 中，PDF 文件格式的帮助文档的内容要多于超文本格式的文档，更是多于纸版的手册。所以，在必要的情况下，可以将部分 PDF 格式的文档打印出来作为手册保存。

MATLAB 6.5 的 PDF 格式文档可以在 MATLAB 产品光盘的第三张光盘上直接察看到，阅读这些文档，需要安装 Adobe Acrobat Reader 4.0 以上版本的阅读器。

尽管 MATLAB 的帮助文档比较详实、规范，用户在使用 MATLAB 的过程中，不可避免地还是会遇到一些问题，这个时候可以使用 MATLAB 的网上资源。利用 MATLAB 图形用户界面上的“Web”菜单下的指令可以直接访问 MATLAB 的网上资源，MATLAB 的“Web”菜单如图 1-30 所示。

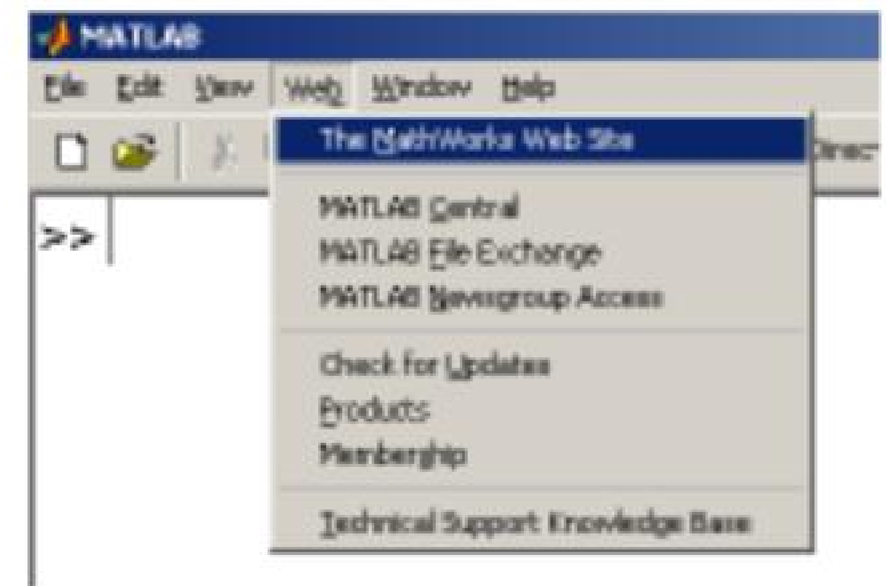


图 1-30　MATLAB 的 Web 菜单。

在所有的 Mathworks 公司提供的网上资源中，笔者推荐大家登录 MATLAB Central，在该网站上不仅可以查阅 MATLAB 的实用信息、在新闻组中提问，还可以通过 File Exchane 功能从网站上下载大量的用户实例，这些例子不仅比 MATLAB 自带的 Demos 要新颖，而且还更贴近实际的工程应用。

其实 MATLAB 在互联网上的资源非常丰富，不仅在 Mathworks 公司的主页上可以找到很多有用的信息，在国内的网站上也有一定规模的信息资源，特别是在北京九州恒润科技公司的主页上，拥有一个以 MATLAB 应用为主的论坛，大家在使用 MATLAB 的过程中若是遇到了问题，可以直接在该论坛上发表自己的问题，九州恒润公司的技术人员会很快为大家解决相应的问题。

另外还可以向 Mathworks 公司的技术人员通过 E-mail 进行技术问题的询问，不过在询问问题的同时需要提供用户产品的信息，这些信息是可以通过在 MATLAB 命令行中键入 ver 指令来获取的，将出现在命令行窗口中的 MATLAB License Number 的内容提供给 Mathworks 公司即可。

1.7.3　操作帮助的函数

MATLAB 还提供了一些函数用于显示帮助信息和操作帮助浏览器，如表 1-4 所示。

表 **1-4**　帮 助 函 数

函　数	说　　明
Help	在 MATLAB 命令行中显示在线帮助
Helpwin	在帮助浏览器中显示在线帮助
Helpbrowser	打开帮助浏览器，并显示超文本帮助文档
Helpdesk	与函数 helpbrowser 功能一致，在早期版本的 MATLAB 中可以打开帮助界面
Doc	打开帮助浏览器，并显示指定的内容
Docroot	帮助文档存在的根目录
Demo	打开帮助浏览器并显示 Demos 标签页
Dbtype	显示 M 文件内容，同时包括文件代码行号
Look or	搜索 H　帮助行
eb	打开帮助浏览器并显示指定的超文本链接内容

这些函数中最常用的是 help 函数和 doc 函数，两者命令行语法基本相同，只不过 doc 函数将打开 MATLAB 的帮助浏览器并显示相应函数的超文本帮助文档。其他函数的具体用法就不再一一叙述了，有兴趣的读者可以查阅帮助文档或者在线帮助。

1.8　本 章 小 结

本章是学习 MATLAB 软件的基础章节。在本章首先介绍了 MATLAB 的产品体系，然后着重介绍了 MATLAB 的各种桌面工具的使用方法。通过本章的学习，读者应该能够对 MATLAB 的产品，以及 MATLAB 的基本使用方法有所了解，为学习后面的章节，或者进一步学习其他内容打下良好的基础。

在所有 MATLAB 桌面工具中，用户使用频率最高的就是 MATLAB 的命令行窗口，通过该窗口几乎能够实现 MATLAB 的所有功能，甚至一些常用的 MATLAB 图形化工具也能够通过命令行窗口的指令来调用执行。

在其他的桌面工具中，最重要的工具就是 MATLAB 的帮助系统，学会使用帮助系统是掌握 MATLAB 非常重要的一步。MATLAB 的帮助系统由在线帮助和窗口帮助两部分组成，两种帮助都有自己的特色。一般的来说，通过在线帮助获得信息最快捷，而通过窗口帮助得到的信息最全面。笔者希望读者在以后学习使用 MATLAB 的过程中，出现的问题都通过查阅帮助文档来加以解决，因为，只有这样才能够得到真正的提高。

在本书后面的章节中将陆续介绍 MATLAB 其他的图形用户界面工具，例如工作空间浏览器(Workspace)、M 文件性能剖析器(Profiler)等。

第二章　矩阵和数组

MATLAB 既是一种进行科学计算、数据处理的环境，又是一种编程开发的环境。MATLAB 提供了一种计算机高级编程语言——M 语言，这种编程语言是用来扩展 MATLAB 功能的强有力的工具。为了使用这种编程语言，MATLAB 还相应地提供了不同类型的数据，所以掌握和了解 MATLAB 的数据类型是掌握和学习 M 语言的前提。

众所周知，作为一种科学计算软件，MATLAB 专门以矩阵作为基本的运算单位，而从计算机编程语言的角度出发，为了能够和 C 语言等高级语言保持一定的相似性，MATLAB 的矩阵在 M 语言中使用数组的形式来表示，而且 MATLAB 还提供了关于数组和矩阵不同的运算方法。所以使用 MATLAB 也必须掌握基本的矩阵运算和数组运算方法。

本章讲述的主要内容如下：

- 矩阵和向量；
- 矩阵运算；
- 数组运算；
- 稀疏矩阵；
- 多维数组。

2.1　概　　述

一种完整的计算机应用语言应该提供对数据的描述和对数据的操作。作为一种高级语言——M 语言，同样提供了对各种类型数据的描述和操作的能力。在 M 语言中，最常用的数据类型表现手段和形式就是变量和常量。

由于 MATLAB 软件自身的特点，它是一种以科学计算为基础的软件，因此 M 语言的基本处理单位是数值矩阵或者数值向量，在 M 语言中统一将矩阵或者向量称之为数组。因此掌握一些基本的有关矩阵、向量和数组操作的基本知识就成为了掌握 MATLAB 软件的前提条件。在本小节将简要地回顾一下有关向量、矩阵和数组的概念。如果读者对本小节的内容非常熟悉，则可以快速浏览本小节。

1．变量和常量

变量和常量是编程语言中数据类型的表现手段和形式，所以从 M 语言的角度而言，掌握变量和常量的概念也是掌握 M 语言编程的基础。

所谓变量，就是指在程序运行过程中需要改变数值的量，每一个变量都具有一个名字，变量将在内存中占据一定的空间，以便在程序运行的过程中保存其数值。M 语言和 C 语言类似，对变量的命名有相应的要求：变量必须以字母开头，后面可以是字母、数字或者下

划线的组合。尽管在编写程序的时候可以使用任意数量的字符表示变量名，但是 MATLAB 仅仅识别前面的 N 个字符，在不同的操作系统下可以识别的字符个数不尽相同，可以使用 namelengthmax 函数察看相应的定义。

所谓常量就是在程序运行的过程中不需要改变数值的量，例如，在求圆周周长或者圆的面积的时候，需要一个常量 π，它的值近似是 3.1415927，常量也具有相应的名字，其定义方法和变量一样。M 语言中的常量不像 C 语言中的常量，一般地在 M 语言中并不存在常量的定义，任何常量和变量都可以修改其数值，只不过在 MATLAB 中提供了一些常用的常数作为常量，这一点请读者注意。

2．数组

一般的，数组是有序数据的集合，在大多数编程语言中，数组的每一个成员(元素)都属于同一种数据类型，它们使用同一个数组名称和不同的下标来惟一确定数组中的成员(元素)。其中，下标是指数组元素在数组中的序号。

对于 MATLAB 而言，大多数数据类型的数组每一个元素都是同一个数据类型的元素，而对于其特殊的元胞数组则不然。

和一般的编程语言类似，M 语言的数组也有一维、二维和多维数组的区别。而在 MATLAB 中一般不存在数组的数组，除非在 M 语言中使用 Java 数据对象。

注意：

有关元胞数组的概念将在本书的第三章中介绍，在 MATLAB 中使用 Java 数据对象的方法请参阅《MATLAB 外部接口编程》一书，或者阅读 MATLAB 相关的帮助文档。

3．向量

从编程语言的角度上看，向量其实就是一维数组，然而从数学的角度上看，向量就是 1×N 或者 N×1 的矩阵，即行向量或列向量，即

$B = \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \\ \vdots \\ b_{n,1} \end{bmatrix}$ 和 $B = \lfloor b_{1,1} \;\; b_{1,2} \;\; b_{1,3} \;\; \cdots \;\; b_{1,n} \rfloor$ 都是一维数组，但是从数学的角度上看，分别被称为列向量和行向量。

MATLAB 的基本运算单位就是矩阵和向量，而 M 语言本身就是以向量化运算为基础的编程语言，正因为有如此特点，使用 M 语言成为了目前最流行的算法开发和验证的原型语言。

4．矩阵

在 MATLAB 中，矩阵的概念就是线性代数中定义的矩阵的概念——矩阵是用一对圆括号或者方括号括起来，符合一定规则的数学对象。例如：

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{23} & b_{33} \end{bmatrix}$$

就是一个三行三列的方阵。

随着线性代数理论的发展，矩阵和向量的运算在工程领域内越来越普遍，因此对于矩阵运算的实现成为诸多计算机软件工程师必须解决的问题。

对于编程语言，矩阵就是二维的数组，而由于一般的编程语言仅能处理单个元素的运算，对于矩阵或者向量的处理，很难按照线性代数的运算法则，将其作为一个整体来处理，从而增加了程序员的工作量，也降低了程序的执行效率和开发周期。于是，诞生了很多专门用于处理矩阵运算的软件包和算法包，其中，MATLAB 软件就是从 EISPACK 和 LINPAC 两个线性代数软件包的基础上发展起来的。

由于篇幅有限，在本章不可能一一回顾线性代数的基本数学知识，有兴趣的读者请自行翻阅参考书籍。应该说，掌握一定的线性代数知识是掌握和精通 MATLAB 软件的理论基础。

2.2 创 建 向 量

从编程语言的角度上看，向量也就是一维数组。在 MATLAB 中创建向量可以使用不同的方法，最直接也最简单的方法就是逐个输入向量的元素，见例子 2-1。

例子 2-1　利用逐个输入元素的方法在 **MATLAB** 中创建向量。

在命令行窗口中键入：

```
>> x = [1 3 pi 3+5i]
x =
   1.0000          3.0000          3.1416          3.0000 + 5.0000i
>> whos
   Name        Size                   Bytes   Class
   x           1x4                       64   double array (complex)
Grand total is 4 elements using 64 bytes
```

在例子 2-1 中，逐个输入向量的元素，创建了一个向量 x。其中需要注意，pi 为 MATLAB 中内建的常量，表示常量 p。

使用逐个元素输入的方法创建向量的时候，彼此元素之间可以使用空格或者逗号“，”作为间隔符。

第二种创建向量的方法可以利用运算符“：”，参阅例子 2-2。

例子 2-2　利用冒号运算符创建向量。

在命令行窗口中键入：

```
>> x = 1:10
x =
    1     2     3     4     5     6     7     8     9    10
>> whos
   Name        Size                   Bytes   Class
   x           1x10                      80   double array
Grand total is 10 elements using 80 bytes
```

在例子 2-2 中使用冒号运算符创建了具有 10 个元素的向量。利用冒号运算符创建向量的基本语法如下：

X = J:INC:K

其中

- J 为向量的第一个元素，而 K 为向量的最后一个元素，INC 为向量元素递增的步长；
- J、INC 和 K 之间必须用“：”间隔；
- 若在表达式中忽略 INC(如例子 2-2 所示)，则默认的递增步长为 1；
- INC 可以为正数也可以为负数，若 INC 为正数，则必须 J<K，若 INC 为负数，则必须 J>K，否则创建的为空向量。

例子 2-3　使用冒号运算符创建向量。

在命令行窗口中键入：

```
>> x = 1:0.01:1.1
x =
    Columns 1 through 6
      1.0000      1.0100      1.0200      1.0300      1.0400      1.0500
    Columns 7 through 11
      1.0600      1.0700      1.0800      1.0900      1.1000
```

创建向量的第三种方法是使用函数 linspace 和 logspace。

linspace 是用来创建线性间隔向量的函数，函数 linspace 的基本语法为

x = linspace(x1,x2,n)

其中

- x1 为向量的第一个元素，x2 为向量的最后一个元素，n 为向量具有的元素个数，函数将根据 n 的数值平均计算元素之间的间隔，间隔的计算公式为 $\frac{x2-x1}{n-1}$；
- 若在表达式中忽略参数 n，则系统默认地将向量设置为 100 个元素。

函数的具体使用方法参见例子 2-4。

例子 2-4　使用 **linspace** 函数创建向量。

```
>> x = linspace(1,2,5)
x =
     1.0000      1.2500      1.5000      1.7500      2.0000
```

在本例子中，使用 linspace 函数创建了一个具有五个元素的向量，而元素之间彼此的间隔为 $\frac{2-1}{5-1}=0.25$ 。

另外一个函数 logspace 被用来创建对数空间的向量，该函数的基本语法为

x = logspace(x1,x2,n)

其中：

- 该函数创建的向量第一个元素值为 x1，而最后一个元素的数值为 x2，n 为向量的元素个数，元素彼此之间的间隔按照对数空间的间隔设置；
- 若在表达式中忽略参数 n，则参数默认地将向量设置为 50 个元素。

该函数的使用参见例子 2-5。

例子 2-5　使用 **logspace** 函数创建向量。

在 MATLAB 的命令行窗口中键入下面的指令：

```
>> x = logspace(1,3,3)
x =
      10        100       1000
```

上面创建的向量都是行向量，也就是说，创建的都是一行 n 列的二维数组(n 表示元素的个数)，如果需要创建列向量，即 n 行一列的二维数组(n 表示元素的个数)，则需要使用分号作为元素与元素之间的间隔或者直接使用转置运算符“'”，参见例子 2-6。

例子 2-6　创建列向量。

```
>> %直接输入元素的方法创建列向量
>> A = [1;2;3;4;5;6]
A =
      1
      2
      3
      4
      5
      6
>> %使用转置的方法创建列向量
>> B = (1:6)'
B =
      1
      2
      3
      4
      5
      6
>> whos
  Name        Size                    Bytes  Class
  A           6x1                        48  double array
  B           6x1                        48  double array
Grand total is 12 elements using 96 bytes
```

在例子 2-6 中使用两种方法创建了列向量，其实这两种方法创建的向量是完全一致的。

2.3　创 建 矩 阵

一般的矩阵具有 m 行 n 列，所以在编程语言中，矩阵和二维数组一般指的是同一个概念。在 M 语言中，矩阵的元素可以为任意 MATLAB 数据类型的数值或者对象。创建矩阵

的方法也有多种，不仅可以直接输入元素，还可以使用 MATLAB 的数组编辑器编辑矩阵的元素。

2.3.1　直接输入法

直接输入矩阵元素创建矩阵的方法适合创建元素和行、列数较少的矩阵，首先察看例子 2-7。

例子 2-7　用直接输入矩阵元素的方法创建矩阵。

在 MATLAB 的命令行窗口中键入下面的指令：

```
>> A = [1 2 3;4 5 6;7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> whos
  Name      Size                    Bytes  Class
  A         3x3                        72  double array
Grand total is 9 elements using 72 bytes
```

在上面的例子中创建了一个 3×3 的矩阵，在创建矩阵的时候，需要注意：

- 整个矩阵的元素必须在“[]”中键入；
- 矩阵的元素行与行之间需要使用分号“；”间隔，也可以在需要分行的地方用回车键间隔；
- 矩阵的元素之间可以使用逗号“，”或者空格间隔。

其实创建上面的矩阵时还可以这么做

```
>> B = [1:3;4:6;7:9]
B =
     1     2     3
     4     5     6
     7     8     9
```

可以将矩阵的每一行或者每一列看作一个向量，矩阵就是由若干向量组合而成的。

2.3.2　数组编辑器

前文的例子中使用 whos 指令来察看当前 MATLAB 会话保存在工作空间内存中的各种变量，此外，还可以使用工作空间浏览器察看工作空间中包含的各种变量，图 2-1 所示为执行了例子 2-7 之后的工作空间浏览器。

在工作空间浏览器中可以察看当前工作空间中保存的各种数据的信息，利用工作空间浏览器，在相应的变量上单击鼠标右键，通过弹出的快捷菜单(如图 2-2 所示)可以对矩阵或者向量进行编辑，也可以删除、重命名工作空间的变量，还能够完成数据可视化的工作。



图 2-1　MATLAB 的工作空间浏览器

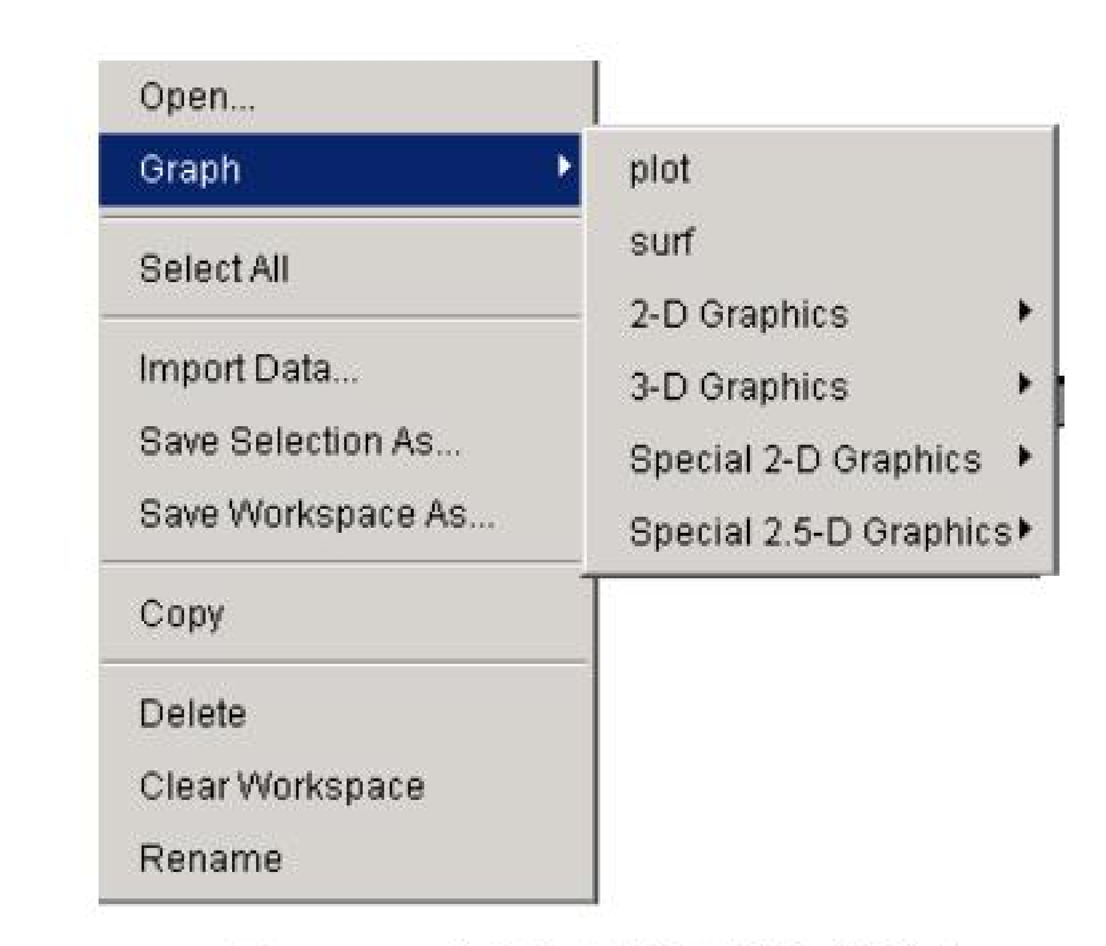


图 2-2　工作空间浏览器的快捷菜单

对矩阵或者向量元素的编辑可以通过数组编辑器来完成，调用数组编辑器有以下几种不同的方法：

第一种，首先选择工作空间浏览器中的变量，然后单击工作空间浏览器工具栏中的按钮，这时系统将运行数组编辑器，同时在编辑器中加载变量。在用鼠标选择工作空间中的变量时，可以按住 Ctrl 键或者 Shift 键选择多个变量，同样也可以使用快捷键 Ctrl+A 选择工作空间中的所有变量。

第二种方法是在工作空间浏览器中直接双击变量，也可以运行数组编辑器，同时在编辑器中加载变量。

第三种方法就是通过工作空间浏览器中的快捷菜单命令 Open 来完成同样的工作。

第四种方法是直接在 MATLAB 命令行窗口中键入如下指令：

```
>> openvar A
```

也可以在数组编辑器中打开变量 A。打开矩阵 A 的数组编辑器界面如图 2-3 所示。

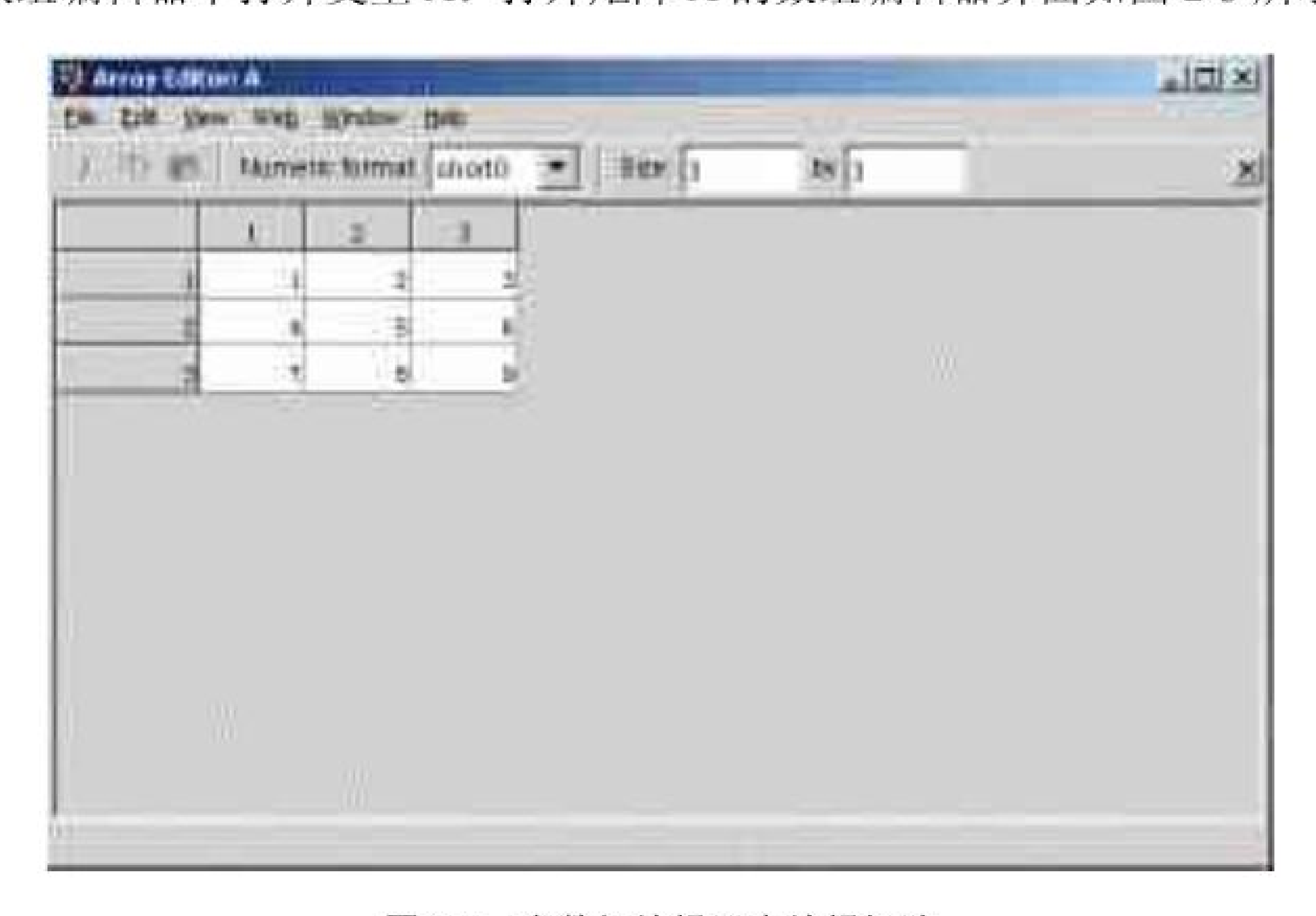


图 2-3　在数组编辑器中编辑矩阵

注意：

数组编辑器仅能编辑、修改向量或者矩阵，对于多维数组，数组编辑器仅能察看数组的内容，不能修改多维数组的元素，同样，也不能修改二维以上(含二维)的字符串数据。详细的细节察看本章后面关于字符串和多维数组的章节内容介绍。

前文提及，在命令行窗口中直接输入元素较多的向量或者矩阵时比较麻烦，所以，可以利用数组编辑器完成大矩阵的编辑，具体方法如下：

首先在命令行窗口中创建一个新的变量，可以为这个变量赋任意的数值，例如在 MATLAB 的命令行窗口中键入：A=1。

然后通过工作空间浏览器打开数组编辑器，并在数组编辑器中加载相应的变量。

在数组编辑器的工具栏中，分别修改矩阵的行数和列数，例如设置矩阵的行、列数分别为 14 行、15 列，则数组编辑器将自动扩充矩阵，将未定义的元素赋初值为 0，这时就可以双击任意元素来修改矩阵的元素值了，如图 2-4 所示，即逐个修改必要的元素，完成矩阵的定义。

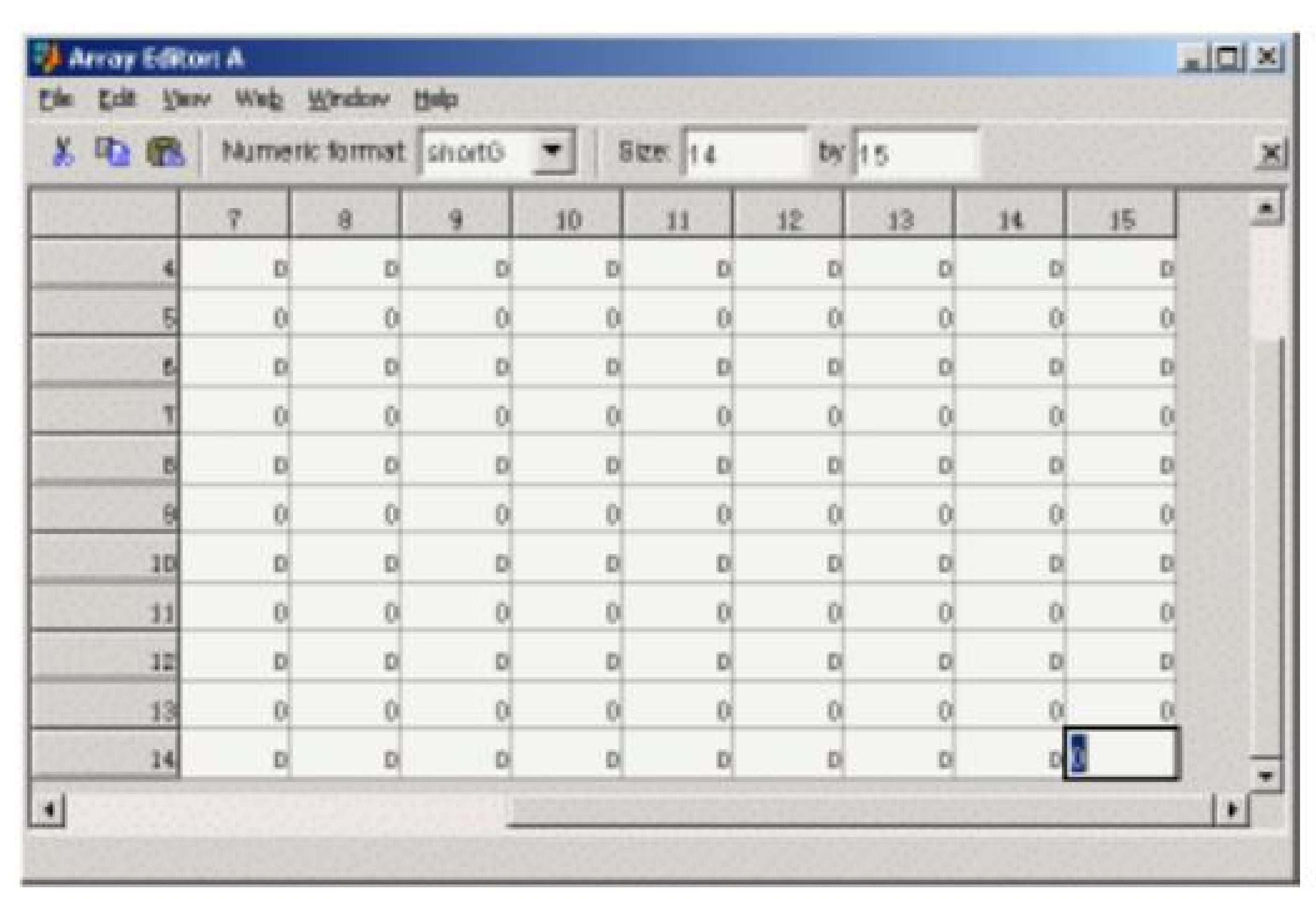


图 2-4　利用数组编辑器定义矩阵

在数组编辑器中可以使用多种格式来显示数组中的数据，这些显示数据的格式和在第一章中介绍 format 指令时介绍的数据格式完全一致。在图 2-5 中演示了利用 bank 格式显示数据的效果。

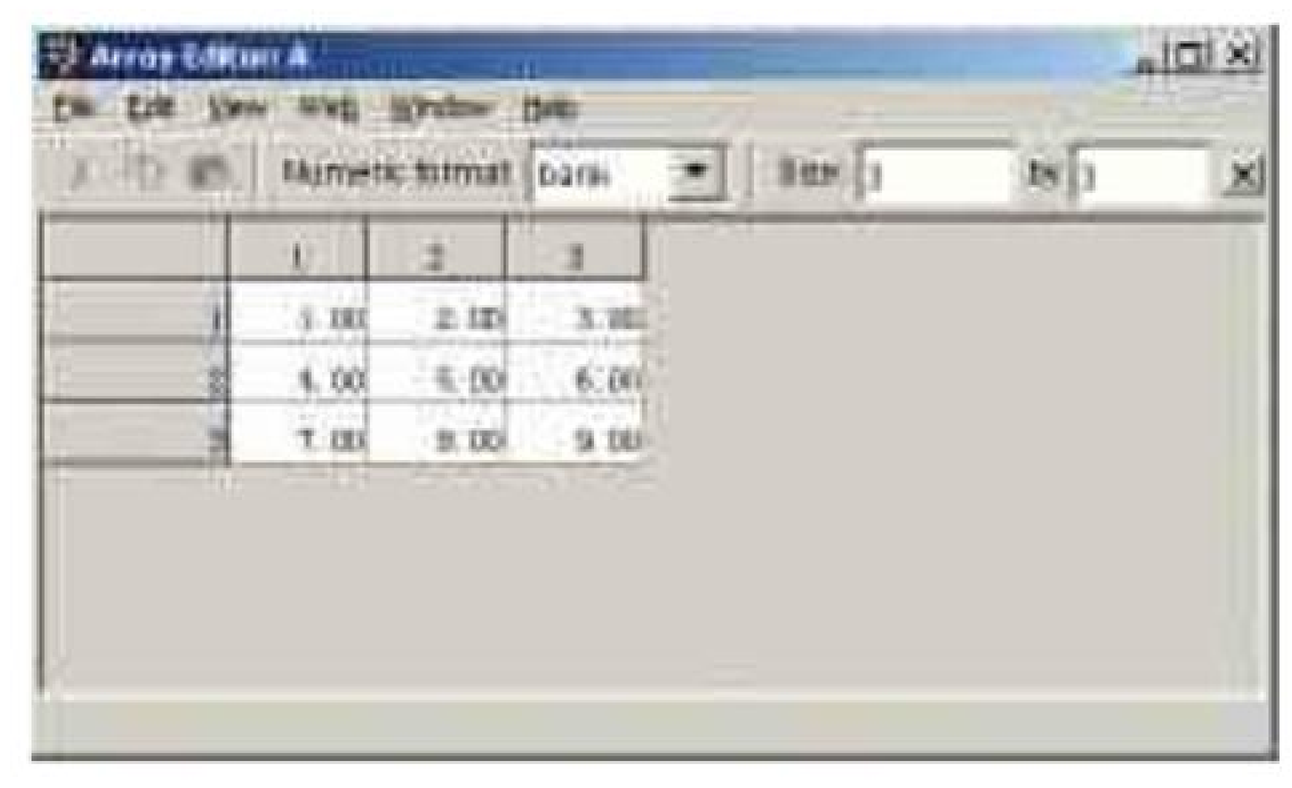


图 2-5　以 bank 格式显示数据

此外，还有一种创建大矩阵的方法就是利用 M 语言的脚本文件，关于脚本文件创建矩阵的方法在第四章中有详细讲述。

2.4 索　　引

前面两个小节中讲述了创建矩阵和向量的方法，在本小节中将详细介绍访问和操作向量或者矩阵元素的方法，这就是利用矩阵或者向量元素的索引来完成相应的操作。

注意：

MATLAB 的矩阵或者数组的索引起始数值为 1，这一点和 C 语言不同，C 语言的数组索引下标的起始数值为 0。

2.4.1 向量元素的访问

访问向量的元素只要使用相应元素的索引即可，请参阅下面的例子 2-8。在例子 2-8 中操作对象是一个向量，该向量为 A = [1 2 3 4 5 6 7 8 9 0]。

例子 2-8　访问向量中的元素。

在 MATLAB 的命令行窗口中键入下面的指令：

```
>> %访问向量的第三个元素
>> A(3)
ans =
      3
>> %访问向量的第一、三、七个元素
>> A([1 3 7])
ans =
      1     3     7
>> %访问向量的第一、三、五个元素
>> A([1:3:5])
ans =
      1     3     5
>> %访问向量的最后四个元素
>> A([end- 3:end])
ans =
      7     8     9     0
>> %重复访问向量中的元素
>> A([1:5,5: - 1:1])
ans =
      1     2     3     4     5     5     4     3     2     1
```

说明：

■ 访问向量元素的结果是创建新的向量。

■ 访问向量的元素直接给出元素在向量中的序号，元素的序号不仅可以是单一的整数，还可以是元素序号组成的向量，如例子 2-8 中的各种操作。
■ 关键字 end 在访问向量元素时，表示向量中最后一个元素的序号。
■ 访问向量元素时，序号的数值必须介于数值 1～end 之间。

可以通过访问元素的方法，对具体的元素赋值，参见例子 2-9。

例子 2-9　对向量的元素进行赋值。

在 MATLAB 命令行窗口中键入下面的指令：

```
>> %对向量的第三个元素赋值
>> A(3) = - 3
A =
     1     2    - 3     4     5     6     7     8     9     0
>> %对向量中不存在的数据赋值
>> A(15) = - 15
A =
  Columns 1 through 10
     1     2    - 3     4     5     6     7     8     9     0
  Columns 11 through 15
     0     0     0     0   - 15
```

说明：

在例子 2-9 中，对向量的第 15 个元素赋值，在赋值之前向量的第 11~15 个元素不存在，但是在赋值之后，将自动创建这些元素，并且为没有明确赋值的元素赋默认值 0，这就是 MATLAB 的数据自动扩充和初始化机制。

2.4.2　矩阵元素的访问

访问矩阵的元素也需要使用矩阵元素的索引，不过具有两种方式，第一种方式是使用矩阵元素的行列全下标形式，第二种方法是使用矩阵元素的单下标形式，参阅例子 2-10。

例子 2-10　访问矩阵的元素。

MATLAB 工作空间中具有一个 5×5 的矩阵，该矩阵是五阶的幻方，通过命令行获取矩阵的第二行、第四列的元素，于是在 MATLAB 命令行窗口中键入下面的指令：

```
>> %创建矩阵
>> A = magic(5)
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
>> %使用全下标的形式访问元素
```

```
>> A(2,4)
ans =
     14
>> %使用单下标的形式访问元素
>> A(17)
ans =
     14
```

说明：

- 使用了 MATLAB 函数 magic 创建幻方。所谓幻方，就是 n 阶的方阵，该方阵的行元素和列元素的和都相等。
- 使用全下标的形式访问矩阵元素的方法简单、直接，同线性代数的矩阵元素的概念一一对应。
- 矩阵元素的单下标是矩阵元素在内存中存储的序列号，一般地，同一个矩阵的元素存储在连续的内存单元中。
- 矩阵元素的单下标与全下标之间的转换关系如下，以 $m\times n$ 的矩阵为例，该矩阵的第 i 行第 j 列的元素全下标表示为单下标 $l=(j-1)\times m+i$。

注意：

MATLAB 的矩阵元素的排列以列元素优先，这一点同 FORTRAN 语言的二维数组元素的排列方法一致，与 C 语言的二维数组元素的排列不同，C 语言的二维数组元素排列以行元素优先。

为了方便全下标和单下标之间的转换，MATLAB 提供了两个函数分别完成两者之间的相互转化：

- sub2ind：根据全下标计算单下标。
- ind2sub：根据单下标计算全下标。

关于上述的两个函数将在多维数组一节详细讲述。

在表 2-1 中总结了使用索引访问矩阵元素的方法。

表 **2-1**　使用索引访问矩阵元素的方法

矩阵元素的访问	说　明
A(i,j)	访问矩阵 A 的第 i 行第 j 列上的元素，其中 i 和 j 为标量
A(I,J)	访问由向量 I 和 J 指定的矩阵 A 中的元素
A(i,:)	访问矩阵 A 中第 i 行的所有元素
A(:,j)	访问矩阵 A 中第 j 列的所有元素
A(:)	访问矩阵 A 的所有元素，将矩阵看作一个向量
A(l)	使用单下标的方式访问矩阵元素，其中 l 为标量
A(L)	访问由向量 L 指定的矩阵 A 的元素，向量 L 中的元素为矩阵元素的单下标数值

注意：

表 2-1 中说明了冒号“:”运算符的运用。在索引矩阵或者数组的元素时，如果直接使用冒号运算符且不给任何的参数，则表示选择该行或者列，或者维(多维数组时)中的所有的元素。

例子 2-11　用不同的方法访问矩阵的元素。

在 MATLAB 命令行中键入下面的指令：

```
>> % 创建矩阵
>> A = 1:25;
>> A = reshape(A,5,5)
A =
      1     6    11    16    21
      2     7    12    17    22
      3     8    13    18    23
      4     9    14    19    24
      5    10    15    20    25
>> %访问矩阵的第三行第一列元素
>> A(3,1)或 A(3)
ans =
      3
>> % 访问矩阵第三行的所有元素
>> A(3,:)
ans =
      3     8    13    18    23
>> %访问矩阵第四列的所有元素
>> A(:,4)
ans =
     16
     17
     18
     19
     20
>> %访问矩阵的最后一行元素
>> A(end,:)
ans =
      5    10    15    20    25
>> %获取矩阵的子矩阵
>> I = [1 3 5];J = [2 4];
>> A(I,J)
ans =
      6    16
      8    18
     10    20
```

在本例子中，使用了 reshape 函数将向量重构成为了一个矩阵，然后分别使用不同的方法获取了矩阵的元素，特别在最后的操作中，获取了矩阵 A 的第一、三、五行第二、四列上的元素，创建子矩阵。

关于 reshape 函数将在本章后面的章节中进行详细介绍，读者也可以参阅 MATLAB 的相关帮助文档。

还有一种访问矩阵或者数组元素的方法是利用布尔类型的索引，关于关系运算将在第四章详细讲述。

2.5 基 本 运 算

作为数学计算软件的 MATLAB 最基本的运算是通过矩阵来完成的，矩阵或者向量几乎作为 MATLAB 所有运算的基础。在本小节，将介绍 MATLAB 中关于矩阵的基本运算，这里主要是一些函数或者基本数学运算的规则。另外，在 MATLAB 中还有一类运算函数和指令用来处理有关数组的运算，本小节中将讨论这类函数。

2.5.1 矩阵生成函数

MATLAB 包含若干函数，这些可以用来生成某些矩阵，参见表 2-2。

表 **2-2** **MATLAB** 的矩阵生成函数

函　数	说　　明
zeros	产生元素全为 0 的矩阵
ones	产生元素全为 1 的矩阵
eye	产生单位矩阵
rand	产生均匀分布的随机数矩阵，数值范围(0,1)
randn	产生均值为 0，方差为 1 的正态分布随机数矩阵
diag	获取矩阵的对角线元素，也可生成对角矩阵
tril	产生下三角矩阵
triu	产生上三角矩阵
pascal	产生帕斯卡矩阵
magic	产生幻方阵

例子 2-12　矩阵生成函数的示例。

在 MATLAB 命令行中键入下面的指令：

```
>> %创建三阶帕斯卡矩阵
>> A = pascal(3)
A =
     1     1     1
     1     2     3
     1     3     6
>> %从矩阵 A 生成下三角矩阵
>> tril(A)
```

```
ans =
     1     0     0
     1     2     0
     1     3     6
>> %获取矩阵 A 的对角线元素
>> diag(A)
ans =
     1
     2
     6
>> %利用向量生成对角矩阵
>> diag(ans)
ans =
     1     0     0
     0     2     0
     0     0     6
```

在表格 2-2 中所罗列的各种矩阵创建函数不仅可以用来创建二维的矩阵，还可以用来创建多维数组。创建多维数组的方法将在后面的章节中进行介绍。

另外，在这些函数中，比较重要的是函数 zeros 和 ones，这两个函数经常在编写 M 语言程序的时候用来初始化大矩阵，以提高程序的执行效率。

2.5.2　基本矩阵运算

针对矩阵的运算 MATLAB 提供了若干函数和基本的运算规则，这些规则和函数都分别和线性代数的基本概念和运算规则对应。矩阵的基本运算参见表 2-3。

表 **2-3**　矩阵的基本运算

运算命令	说　明
A'	矩阵转置
A^n	矩阵求幂，n 可以为任意实数
A*B	矩阵相乘
A/B	矩阵右除
A\B	矩阵左除
A+B	矩阵加法
A- B	矩阵相减
inv	矩阵求逆，注意不是所有的矩阵都有逆矩阵
det	求方阵的行列式
rank	求矩阵的秩
eig	求矩阵的特征向量和特征值
svd	对矩阵进行奇异值分解
Norm	求矩阵的范数

提示：

在 MATLAB 中，获取矩阵(线性代数)的运算函数列表请在 MATLAB 命令行窗口中键入

如下命令：

```
>> help matfun
```

在 MATLAB 命令行窗口中将显示相应的函数列表：

```
Matrix functions - numerical linear algebra.
Matrix analysis.
    norm        - Matrix or vector norm.
    normest     - Estimate the matrix 2-norm.
⋮
```

一般的 MATLAB 函数都可以针对矩阵进行运算，但是在前面的 help 命令行中显示的函数是专门针对矩阵和线性代数运算的函数。

有关每一个函数的具体用法，请参阅相应的帮助文档或者 MATALB 在线帮助。

例子 2-13　矩阵的基本运算示例——求解方程组。

$$\begin{cases} -x_1 + x_2 + 2x_3 = 2 \\ 3x_1 - x_2 + x_3 = 6 \\ -x_1 + 3x_2 + 4x_3 = 4 \end{cases}$$

这类问题可以直接通过矩阵运算解决。在 MATLAB 命令行窗口中键入下面的指令：

```
>> %创建线性方程组的系数矩阵和向量
>> A = [- 1 1 2; 3 - 1 1; - 1 3 4];
>> b = [2;6;4];
>> %求解方程，使用矩阵求逆的方法
>> x = inv(A)*b
x =
    1.0000
   - 1.0000
    2.0000
>> %求解方程，使用矩阵左除运算
>> x = A\b
x =
    1.0000
   - 1.0000
    2.0000
```

从例子 2-13 可以看出以矩阵为基本运算单位进行数值运算的优势。对于 MATLAB，矩阵的基本运算都可以用一种最简单、直观的表达式完成，像这种利用向量或者矩阵的运算，不仅可以简化代码，还能够提高 MATLAB 的 M 语言的运行速度。

矩阵的运算同时也包含了矩阵和标量之间的运算，MATLAB 在处理这种运算的时候，首先对标量进行扩充，例如：

```
>>  w=[1 2;3 4] + 5
w =
```

```
        6       7
        8       9
```

该指令行实际的执行过程如下：

w=[1 2;3 4] + 5

$$=\begin{bmatrix}1&2\\3&4\end{bmatrix}+5=\begin{bmatrix}1&2\\3&4\end{bmatrix}+\begin{bmatrix}5&5\\5&5\end{bmatrix}$$

$$=\begin{bmatrix}6&7\\8&9\end{bmatrix}$$

2.5.3　基本数组运算

在 MATLAB 中，除了基本的线性代数运算以外，大多数的矩阵运算都适用于数组运算，但是有部分运算不同，本小节将介绍其中的基本运算指令和方法。

1．数组转置

数组转置的操作符是在矩阵转置操作符前加符号“.”，见例子 2-14。

例子 2-14　数组转置操作。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> %创建矩阵
>> A = ones(2,3); A(:)=1:6
A =
        1       3       5
        2       4       6
>> %矩阵转置
>> A'.
ans =
        1       2
        3       4
        5       6
>> %数组转置
>> A'.
ans =
        1       2
        3       4
        5       6
>> %复数运算，矩阵 A 成为复数矩阵
>> A = A*I
```

```
A =
          0 + 1.0000i          0 + 3.0000i          0 + 5.0000i
          0 + 2.0000i          0 + 4.0000i          0 + 6.0000i
>> %矩阵转置
>> A'
ans =
          0 - 1.0000i          0 - 2.0000i
          0 - 3.0000i          0 - 4.0000i
          0 - 5.0000i          0 - 6.0000i
>> %数组转置
>> A.'
ans =
          0 + 1.0000i          0 + 2.0000i
          0 + 3.0000i          0 + 4.0000i
          0 + 5.0000i          0 + 6.0000i
```

从例子 2-14 可以看出，对于实数矩阵，矩阵转置和数组转置的计算结果是一致的，但是对于复数矩阵，数组转置和矩阵转置的计算结果就不一致。所以，对于数组转置运算也被称为非共轭转置，矩阵转置运算则被称为共轭转置。

2．数组幂

数组幂运算符就是在矩阵幂运算符前加上符号“.”，见例子 2-15。

例子 2-15　数组幂运算。

在 MATLAB 命令行中，键入下面的指令：

```
>> %本例子中使用的矩阵
>> A
A =
          0 + 1.0000i          0 + 3.0000i          0 + 5.0000i
          0 + 2.0000i          0 + 4.0000i          0 + 6.0000i
>> %矩阵幂运算
>> A^3
??? Error using ==> ^
Matrix must be square.
>> %数组幂运算
>> A.^3
ans =
   1.0e+002 *

          0 - 0.0100i          0 - 0.2700i          0 - 1.2500i
          0 - 0.0800i          0 - 0.6400i          0 - 2.1600i
```

在例子 2-15 中，可以看出矩阵的幂运算仅对方阵有效，所以在求矩阵 A 的立方时，MATLAB 报告了错误，而数组幂运算则对任意矩阵均有效，数组幂运算是将矩阵的对应元素进行幂运算。

3．数组乘法

和前面两种运算类似，数组乘法运算符是在矩阵乘法运算符前加上符号“.”，见例子 2-16。

例子 2-16　数组乘法示例。

在 MATLAB 命令行中，键入下面的指令：

```
>> %本例子中使用的矩阵
>> A
A =
        0 + 1.0000i        0 + 3.0000i        0 + 5.0000i
        0 + 2.0000i        0 + 4.0000i        0 + 6.0000i
>> %矩阵乘法
>> A*5
ans =
        0 + 5.0000i        0 +15.0000i        0 +25.0000i
        0 +10.0000i        0 +20.0000i        0 +30.0000i
>> %数组乘法
>> A.*5
ans =
        0 + 5.0000i        0 +15.0000i        0 +25.0000i
        0 +10.0000i        0 +20.0000i        0 +30.0000i
>> %矩阵乘法
>> A*A'
ans =
    35    44
    44    56
>> %数组乘法
>> A.*A
ans =
    - 1    - 9    - 25
    - 4   - 16    - 36
```

通过本例子可以看出，在 MATLAB 中，矩阵和标量之间的乘法运算通过矩阵乘法和数组乘法得到的结果是一致的，但是矩阵和矩阵之间的乘法运算则不然，矩阵之间的乘法有一定基本原则，数组乘法与矩阵乘法的运算规则不同，如例子 2-16 中的乘法运算。

最后一种基本数组运算是数组除法，和前面的几种数组运算类似，读者可以自己实践掌握数组除法的运算，本书中就不再赘述了。

2.5.4　基本数学函数

在 MATLAB 中有部分函数可以用来进行基本的数学运算，主要有如下类别：三角函数(见表 2-4)、指数运算函数(见表 2-5)、复数运算函数(见表 2-6)、圆整和求余函数(见表 2-7)。需要注意的是这些函数的参数可以是矩阵也可以是向量或者多维数组，函数在处理参数时，都是按照数组运算的规则来进行的，也就是说对于 $m\times n$ 的矩阵 $A=[a_{ij}]_{m\times n}$，函数 f(?)的运算指 $f(A)=[f(a_{ij})]_{m\times n}$。

表 **2-4**　三 角 函 数

函 数	说　明	函 数	说　明	函 数	说　明
sin	正弦函数	tanh	双曲正切函数	csch	双曲余割函数
sinh	双曲正弦函数	atan	反正切函数	acsc	反余割函数
asin	反正弦函数	atan2	四象限反正切函数	acsch	反双曲余割函数
asinh	反双曲正弦函数	atanh	反双曲正切函数	cot	余切函数
cos	余弦函数	sec	正割函数	coth	双曲余切函数
cosh	双曲余弦函数	sech	双曲正割函数	acot	反余切函数
acos	反余弦函数	asec	反正割函数	acoth	反双曲余切函数
acosh	反双曲余弦函数	asech	双曲反正割函数		
tan	正切函数	csc	余割函数		

表 **2-5**　指数运算函数

函 数	说　明	函 数	说　明
exp	指数函数	realpow	实数幂运算函数
log	自然对数函数	reallog	实数自然对数函数
log10	常用对数函数	realsqrt	实数平方根函数
log2	以 2 为底的对数函数	sqrt	平方根函数
pow2	2 的幂函数	nextpow2	求大于输入参数的第一个 2 的幂

说明：

以 real 开头的函数仅能处理实数，如输入的参数为复数，则 MATLAB 会报错。

函数 nextpow2 是用来计算仅仅比输入参数大的第一个 2 的幂，例如输入参数为 N，则函数的计算结果整数 P 需要满足的条件为 $2^{P}\geqslant \mathrm{abs}(N)\geqslant 2^{P-1}$。

表 **2-6**　复 数 运 算

函 数	说　明	函 数	说　明
abs	求复数的模，若参数为实数则求绝对值	real	求复数的实部
angle	求复数的相角	unwrap	相位角按照 360° 线调整
complex	构造复数	isreal	判断输入参数是否为实数
conj	求复数的共轭复数	cplxpair	复数阵成共轭对形式排列
image	求复数的虚部		

表 **2-7**　圆整和求余函数

函 数	说 明	函 数	说 明
fix	向 0 取整的函数	mod	求模函数
floor	向 $-\infty$ 取整的函数	rem	求余数
ceil	向 $+\infty$ 取整的函数	sign	符号函数
round	向最近的整数取整的函数		

关于表 2-7 中的函数的用法参阅示例 2-17。

例子 2-17　MATLAB 的圆整和求余函数。

在 MATLAB 的命令行中，键入下面的指令：

```
>> fix(- 1.9)
ans =
     - 1
>> floor(- 1.9)
ans =
     - 2
>> round(- 1.9)
ans =
     - 2
>> ceil(- 1.9)
ans =
     - 1
```

上面比较了四种圆整函数处理同一个数据的结果，在使用不同的取整函数时要注意各个函数的特点。其实这四种圆整函数之间的区别主要是进行圆整运算时，趋近的方向不尽相同。例如 fix 函数是将数据向 0 的方向趋近，而 floor 函数是向无穷大的方向上趋近。

```
>> mod(9, - 2)
ans =
     - 1
>> rem(9, - 2)
ans =
     1
```

这里比较了两种取余运算函数的区别，请读者仔细观察两者的不同点，就能够总结出两个函数运行结果的差别和函数运算的不同。读者可以参阅 MATLAB 中这些函数的帮助文档和在线帮助。

2.5.5　矩阵(数组)操作函数

在前面的小节中主要介绍了进行数学运算的 MATLAB 函数，在 MATLAB 中还存在一类函数用来获取矩阵或者数组的信息，以及对数组进行操作，在表 2-8 中列举了较常用的函数。完整的函数列表内容可以在 MATLAB 命令行中键入 help elmat 指令来察看。

表 **2-8**　用于矩阵(数组)操作的常用函数

函　数	说　明
size	获取矩阵的行、列数，对于多维数组，获取数组的各个维的尺寸
length	获取向量长度，若输入参数为矩阵或多维数组，则返回各个维尺寸的最大值
ndims	获取矩阵或者多维数组的维数
numel	获取矩阵或者数组的元素个数
disp	显示矩阵或者字符串内容(有关字符串的内容将在第三章中讲述)
cat	合并不同的矩阵或者数组
reshape	保持矩阵元素的个数不变，修改矩阵的行数和列数
repmat	复制矩阵元素并扩展矩阵
fliplr	交换矩阵左右对称位置上的元素
flipud	交换矩阵上下对称位置上的元素
flipdim	按照指定的方向翻转交换矩阵元素
find	获取矩阵或者数组中非零元素的索引

例子 2-18　reshape 函数的使用示例。

在前面的例子 2-11 中曾经使用过 reshape 函数，这里将详细讨论该函数的使用方法，在 MATLAB 命令行中，键入下面的指令：

```
>> A = 1:8
A =
     1     2     3     4     5     6     7     8
>> B = reshape(A,2,4)
B =
     1     3     5     7
     2     4     6     8
>> C = reshape(B,3,3)
??? Error using ==> reshape
To RESHAPE the number of elements must not change.
```

使用 reshape 函数时需要注意，不能改变矩阵包含元素的个数，所以在第二次使用 reshape 函数时，MATLAB 报告了相应的错误。

例子 2-19　对称交换函数的使用示例。

在 MATLAB 命令行中，键入下面的指令：

```
>> A = reshape(1:9,3,3)
A =
     1     4     7
     2     5     8
     3     6     9
>> fliplr(A)
ans =
     7     4     1
     8     5     2
     9     6     3
```

```
>> flipud(A)
ans =
     3     6     9
     2     5     8
     1     4     7
>> flipdim(A,1)
ans =
     3     6     9
     2     5     8
     1     4     7
>> flipdim(A,2)
ans =
     7     4     1
     8     5     2
     9     6     3
```

说明：

使用 reshape 函数可以比较方便地创建矩阵，这也是创建矩阵的一种方法。

flipdim 函数的第二个参数必须是大于 0 的整数，其中参数为 1 时，效果和 flipud 函数一致，参数为 2 时，效果和 fliplr 函数一致。

在生成比较复杂的矩阵时，可以使用 MATLAB 提供的矩阵扩展方法完成相应矩阵的构造。假设矩阵 A 为三阶方阵，B 为二阶方阵，由矩阵 A 和 B 组合构成五阶方阵 $C=\begin{bmatrix} A & O \\ O & B \end{bmatrix}$，其中 O 为相应的零矩阵，具体的创建方法见例子 2-20。

例子 2-20　创建复杂矩阵。

在 MATLAB 命令行中，键入下面的指令：

```
>> A = reshape(1:9,3,3);
>> B = [1 2 ; 3 4];
>> O = zeros(length(A),length(B))
O =
     0     0
     0     0
     0     0
>> C = [ A O; O' B]
C =
     1     4     7     0     0
     2     5     8     0     0
     3     6     9     0     0
     0     0     0     1     2
     0     0     0     3     4
```

在例子 2-20 中，使用方括号将不同的矩阵合并起来构成了复杂的大矩阵。这里，方括号实际上是矩阵合并的运算符。在方括号中，空格或逗号用于分隔列，而分号用于分隔行。

再看一个使用方括号运算符的例子。

例子 2-21　使用方括号创建复杂矩阵。

在 MATLAB 命令行中，键入下面的指令：

```
>> A = [1 2 ; 3 4];
>> B = [ A , A*2 ; tril(A) , triu(A) ; A*3 ,   A*4]
B =
     1     2     2     4
     3     4     6     8
     1     0     1     2
     3     4     0     4
     3     6     4     8
     9    12    12    16
```

这里利用不同的矩阵运算，通过矩阵合并运算符“[]”将不同的矩阵组合在一起构成了复杂的大矩阵。通过小矩阵的运算来创建大的复杂的矩阵也是创建矩阵的一种手段，在实际的工作中，需要读者根据具体情况灵活地运用创建矩阵的不同方法。

例子 2-22　函数 repmat 的应用示例。

```
>> repmat(magic(2),2,3)
ans =
     1     3     1     3     1     3
     4     2     4     2     4     2
     1     3     1     3     1     3
     4     2     4     2     4     2
```

repmat 函数的基本语法为 repmat(A,M,N)，它的作用是将指定的矩阵 A 复制 M ×N 次，然后创建一个复杂的大矩阵，结果为 $\begin{bmatrix} A & A & \dots & A \\ A & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ A & \dots & \dots & A \end{bmatrix}_{M\times N}$，因此，在例子 2-22 中，将一个简单的二行二列的矩阵进行了六次重复，创建了四行六列的大矩阵。

2.6　稀 疏 矩 阵

矩阵元素的表示方法是计算机数据结构理论中经常讨论的话题，因为在实际工作中，经常遇到这样一类矩阵，这类矩阵中数值为 0 的元素居多，这类矩阵一般被称为稀疏矩阵。如果使用满阵的方式来表示稀疏矩阵，则 0 元素将占用相当的内存空间，特别是在 MATLAB 中，由于 MATLAB 默认的数据类型是双精度类型，每一个双精度类型的数据元素都要占用八个字节的内存空间，当 0 元素很多的时候将占用相当可观的内存空间，因此，在 MATLAB

中，专门提供了稀疏矩阵的表示方法。

例子 2-23 创建稀疏矩阵。

在 MATLAB 命令行窗口中键入下面的指令：

```
>> A = eye(5)
A =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
>> B = sparse(A)
B =
   (1,1)        1
   (2,2)        1
   (3,3)        1
   (4,4)        1
   (5,5)        1
>> whos
  Name      Size                    Bytes  Class
  A         5x5                       200  double array
  B         5x5                        84  double array (sparse)
Grand total is 30 elements using 284 bytes
```

在例子 2-23 中，首先使用 eye 函数创建了五阶的单位矩阵，五阶单位方阵一共有 25 个元素，其中却有 20 个元素是 0，于是使用 sparse 函数将该函数构造成为稀疏矩阵，这样就得到了矩阵 B。

通过 whos 指令可以清晰地比较两个矩阵占用的内存空间，A 矩阵占用了 200 个字节，而 B 矩阵仅占用了 84 个字节。

稀疏矩阵和普通矩阵(满阵)之间可以直接进行运算，例如

```
>> A+B
ans =
     2     0     0     0     0
     0     2     0     0     0
     0     0     2     0     0
     0     0     0     2     0
     0     0     0     0     2
```

这里运算得到的结果是一个满阵，请读者在进行稀疏矩阵运算的时候注意。

MATLAB 中使用“三元组”表示法来表示稀疏矩阵，该表示方法一般由三个向量组成：第一个向量是由矩阵中非零元素组成的向量，其长度一般为 nzmax，即矩阵中所有非零元素的个数；第二个向量是非零元素的行序号，该向量的长度也为 nzmax；第三个向量是非零元

素的列序号，该向量的长度也为 nzmax。

例如对于下面的稀疏矩阵：

$$S=\begin{bmatrix}15 & 0 & 0 & 22 & 0 & -15\\ 0 & 11 & 3 & 0 & 0 & 0\\ 0 & 0 & 0 & -6 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 91 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 28 & 0 & 0 & 0\end{bmatrix}$$

因此，表示矩阵的三个向量分别为

```
>> data = [15 91 11 3 28 22 - 6 - 15];
>> ir = [ 1 5 2 2 6 1 3 1];
>> jc = [1 1 2 3 3 4 4 6];
```

利用上面的三个矩阵和 sparse 函数创建六行六列的稀疏矩阵：

```
>> S = sparse(ir,jc,data,6,6)
S =
   (1,1)        15
   (5,1)        91
   (2,2)        11
   (2,3)         3
   (6,3)        28
   (1,4)        22
   (3,4)        - 6
   (1,6)       - 15
>> %将该矩阵还原成满阵
>> full(S)
ans =
    15     0     0    22     0   - 15
     0    11     3     0     0     0
     0     0     0   - 6     0     0
     0     0     0     0     0     0
    91     0     0     0     0     0
     0     0    28     0     0     0
>> whos
  Name       Size                Bytes  Class
  S          6x6                   124  double array (sparse)
  ans        6x6                   288  double array
  data       1x8                    64  double array
```

```
ir        1x8        64  double array
jc        1x8        64  double array
Grand total is 68 elements using 604 bytes
```

在上面的创建过程中，请读者注意创建的稀疏矩阵占用了 124 个字节的存储空间，而满阵占用了 288 个字节的空间。

注意：

在不同的语言或者数学计算工具软件中，表示稀疏矩阵的方法可能不尽相同，请读者仔细查阅相应软件的帮助文档。

MATLAB 专门提供了若干函数用于稀疏矩阵的运算，在表 2-9 中对这些函数进行了总结。

提示：

在 MATLAB 命令行窗口中键入指令 help sparfun 可以得到稀疏矩阵运算函数的列表。

表 **2-9**　稀疏矩阵的常用函数

创建稀疏矩阵	
speye	创建单位稀疏矩阵
sprand	创建均匀分布的随机数稀疏矩阵
sprandn	创建正态分布的随机数稀疏矩阵
满阵和稀疏矩阵之间的转换	
sparse	创建稀疏矩阵或者将满阵转变为稀疏矩阵
full	将稀疏矩阵转变为满阵
find	获取非零元素的索引向量
稀疏矩阵的操作函数	
nnz	获取矩阵的非零元素的个数
nozeros	获取矩阵的非零元素向量
nzmax	获取矩阵的各个向量的最大长度
spones	将稀疏矩阵中的元素用数字 1 替代
issparse	判断输入参数是否为稀疏矩阵

本小节就不再针对这些函数的具体使用方法作解释了，有关稀疏矩阵以及操作稀疏矩阵的函数的具体使用方法，请读者自行阅读 MATLAB 的帮助文档，而有关稀疏矩阵的概念请参阅有关数据结构的书籍。

2.7　多 维 数 组

和大多数的编程语言类似，MATLAB 也具有多维数组的概念。所谓多维数组，就是全下标表示元素时，下标超过了两个的数组。对于三维数组，习惯性地从矩阵继承而来将数组的第一维称为行、第二维称为列，而对于数组的第三维则将其称为页。三维数组的每一页上的数组必须具有同样的行、列数，否则 MATLAB 将报告相应的错误。更高维的数组相对来说使用的机会较少，所以，本书中以三维数组来说明多维数组的创建和操作。

2.7.1　创建多维数组

多维数组的创建也有多种方法：第一种方法是使用直接赋值的方法来创建；第二种方法是使用部分 MATLAB 的函数创建多维数组。具体的做法结合例子讲述。

例子 2-24　使用直接赋值的方法创建多维数组。

在 MATLAB 的命令行中，键入下面的指令：

```
>> A = pascal(4)
A =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
>> A(:,:,2) = eye(4)
A(:,:,1) =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
A(:,:,2) =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
>> A(:,:,3) = magic(5)
??? Subscripted assignment dimension mismatch.
```

在例子 2-24 中，首先创建了一个矩阵——帕斯卡矩阵，也可以将其看作二维数组，然后，使用全下标的形式创建了三维数组的第二页，第二页上的矩阵是一个单位阵，接着在创建新的一页时使用了五阶方阵作为输入，由于维数不匹配所以系统报错。

例子 2-25　使用直接赋值的方法创建多维数组。

在 MATLAB 的命令行中，键入下面的指令：

```
>> B(3,3,3) = 1
B(:,:,1) =
     0     0     0
     0     0     0
     0     0     0
B(:,:,2) =
     0     0     0
     0     0     0
     0     0     0
```

```
B(:,:,3) =
     0     0     0
     0     0     0
     0     0     1
```

在本例子中通过直接赋初值的方法创建了三维数组，创建时直接对三维数组第三页上的第三行第三列元素进行赋值，MATLAB 将自动扩充数组，并将没有指定数值的数组元素赋初值为 0。

在 2.5.1 小节介绍的部分创建矩阵的函数也可以用来创建多维数组，这里用 rand 函数作为示例。

例子 2-26　使用函数创建多维数组。

在 MATLAB 命令行中，键入下面的指令：

```
>> rand(3,3,3)
ans(:,:,1) =
    0.9501    0.4860    0.4565
    0.2311    0.8913    0.0185
    0.6068    0.7621    0.8214
ans(:,:,2) =
    0.4447    0.9218    0.4057
    0.6154    0.7382    0.9355
    0.7919    0.1763    0.9169
ans(:,:,3) =
    0.4103    0.3529    0.1389
    0.8936    0.8132    0.2028
    0.0579    0.0099    0.1987
```

在本例子中，直接使用 rand 函数创建了一个三维数组，像 ones、zeros 等函数都可以使用这种方式创建多维数组。

除了这些可以直接创建多维数组的函数以外，还可以使用 cat、repmat 等函数构造多维数组，这里用 cat 函数来创建多维数组。

例子 2-27　使用函数 **cat** 构造多维数组。

在 MATLAB 命令行中，键入下面的指令：

```
>> A = magic(3);
>> B = eye(3);
>> C = pascal(3);
>> cat(3,A,B,C)
ans(:,:,1) =
     8     1     6
     3     5     7
     4     9     2
```

```
ans(:,:,2) =
     1     0     0
     0     1     0
     0     0     1
ans(:,:,3) =
     1     1     1
     1     2     3
     1     3     6
```

在本例子中利用 cat 函数将三个三阶方阵组合构成一个三维数组。

有关构造数组的函数的具体使用方法可参阅 MATLAB 的帮助文档或在线帮助。

访问多维数组的元素和访问向量或者矩阵元素的方法一致，不仅可以使用相应的索引作为下标访问多维数组的元素，而且也可以使用单下标的方式来访问。

在前面 2.4.2 小节曾经提及了两个函数可以用于全下标和单下标之间的转换，这两个函数分别为 sub2ind 和 ind2sub，在例子 2-28 中演示了这两个函数的使用方法。

例子 2-28　单下标和全下标之间的转换。

```
>> %创建多维数组
>> A = rand(3,3,4);
>> %全下标向单下标转换
>> sub2ind(size(A),2,3,2)
ans =
    17
>> %单下标向全下标转换
>> ind2sub(size(A),17)
ans =
    17
>> [i,j,k] = ind2sub(size(A),17)
i =
     2
j =
     3
k =
     2
```

通过例子 2-28 可以看到，如果将单下标向全下标转换不给输出参数，则转换得到的结果还是单下标。两个函数具体的定义和其他使用方法请参阅 MATLAB 的帮助文档。

2.7.2　多维数组的操作函数

在 MATLAB 中有一组函数专门用于处理多维数组，见表 2-10。

表 **2-10**　多维数组的操作函数

函　数	说　　明
ndgrid	根据输入的向量产生用于函数和插值运算的多维数组
permute	改变多维数组的维数顺序
ipermute	permute 函数的逆运算
shiftdim	平移多维数组的维数
circshift	循环平移多维数组的行或列元素
squeeze	进行数组降维操作，将多维数组中维数为 1 的页消除

例子 2-28　permute 函数和 ipermute 函数的使用示例。

在 MATLAB 命令行中，键入下面的指令：

```
>> a = rand(2 ,3, 4);
>> size(a)
ans =
      2      3      4
>> b = permute(a,[ 2 1 3]);
>> size(b)
ans =
      3      2      4
>> c = ipermute(b,[2 1 3]);
>> size(c)
ans =
      2      3      4
```

在本例子中，首先创建了一个二行三列四页的三维数组，然后使用 permute 函数将三维数组转变成为了一个三行二列四页的三维数组。ipermute 函数是 permute 函数的逆运算，在例 2-28 中最后得到的三维数组 c 和三维数组 a 是完全一致的。

注意：

permute 函数和 ipermute 函数的第二个参数是一个向量，向量内的元素是多维数组各个维的序号。

例子 2-29　shiftdim 函数的使用示例。

在 MATLAB 命令行中，键入下面的指令：

```
>> a = rand(1,2,3,4,5);
>> size(a)
ans =
      1      2      3      4      5
>> size(shiftdim(a))
ans =
      2      3      4      5
>> size(shiftdim(a,2))
```

```
ans =
     3     4     5     1     2
>> size(shiftdim(a, - 2))
ans =
     1     1     1     2     3     4     5
```

shiftdim 是平移多维数组各个维数据的函数，注意该函数的第二个参数，当该参数取值为正的时候，将数组向左平移，否则向右平移。

例子 2-30　squeeze 函数的使用示例。

在 MATLAB 命令行中，键入下面的指令：

```
>> a = rand(1,2,1,3,1,4,1,5);
>> size(a)
ans =
     1     2     1     3     1     4     1     5
>> size(squeeze(a))
ans =
     2     3     4     5
```

squeeze 函数的作用是将多维数组中尺寸为 1 的页删除，以缩减多维数组的维数，同时保持数组的元素个数不变。

表 2-10 所列出的函数的具体用法请参阅帮助文档或者 MATLAB 的在线帮助。

2.8 本章小结

矩阵的运算是 MATLAB 运算的基础，所以掌握如何在 MATLAB 环境下使用矩阵是掌握 MATLAB 的基础。本章讨论了向量、矩阵、索引和多维数组的概念，重点讲述了在 MATLAB 环境下创建向量、矩阵和多维数组的方法，以及操作矩阵和多维数组的函数的使用方法。另外，在本章还讲述了稀疏矩阵的概念，以及在 MATLAB 中创建稀疏矩阵和操作稀疏矩阵的函数。本章讨论了大量针对数组和矩阵的运算函数和操作函数，给出了一些典型函数的使用示例。读者需要在掌握这些典型函数的基础上，不断实践、练习，掌握各种处理矩阵和数组运算的函数。

第三章　数 据 类 型

在第二章讨论了有关矩阵和数组的创建和操作，在那里所有的数据都使用了 MATLAB 默认的数据类型——双精度类型。和大多数的高级编程语言类似，MATLAB 也提供了各种不同的数据类型用来操作不同的数据。在本章将详细讨论在 MATLAB 中常用的几种数据类型，以及在 MATLAB 中常用的一些数值常量，同时还要讨论操作这些数据类型的函数的用法。

本章讲述的主要内容如下：

- 基本数值类型；
- 逻辑类型；
- 字符串；
- 元胞数组；
- 结构。

3.1　概　　述

MATLAB 的早期版本只有非常简单的二维数组和字符类型的数据，目前的 MATLAB 版本中不仅有多达十几种的基本数据类型，在不同的专业工具箱中还有特殊的数据类型，并且 MATLAB 还支持面向对象的编程技术，支持用户自定义的数据类型。

MATLAB 支持的基本数据类型见图 3-1。



图 3-1　MATLAB 的数据类型

获取 MATLAB 的数据类型列表可以在 MATLAB 命令行窗口中键入 help datatypes 命令。

在图 3-1 中所示的各种数据类型都可以用于创建向量、矩阵或者多维数组。用户自定义的数据类型也是图示的各种数据类型的组合。在不同的 MATLAB 工具箱中具有自定义的数

据类型，例如控制系统工具箱中定义的 LTI 对象，在滤波器设计工具箱中定义的滤波器对象，在符号数学工具箱中定义的符号类型数据等。这些工具箱中包含的特殊数据对象也都使用这些基本的数据类型组合构成。

需要指出一点，MATLAB 的基本数据类型是双精度数据类型和字符类型。MATLAB 的 M 语言和其他高级编程语言不同的是，MATLAB 没有具体的变量或对象声明和定义过程，任何数据类型的变量或对象都可以利用面向对象编程技术中构造函数的方法或者数据类型转换的方法来创建其他数据类型对象和变量。MATLAB 和 Java 语言、C++语言类似，所有数据类型就是相应的类，具有一定的面向对象的特点。MATLAB 的不同数据类型的变量或对象占用的内存空间不尽相同，不同的数据类型的变量或对象也具有不同的操作函数。在本章中，将详细讲述 MATLAB 的基本数值类型、逻辑类型、字符串、元胞数组和结构的使用方法，其中，元胞数组是 MATLAB 中常用的一种独有的数据类型。

提示：

有关不同工具箱中定义的数据类型、面向对象的编程技术和用户自定义的类、函数句柄以及内嵌对象方面的知识，读者可以阅读相关的帮助文档和书籍。

Java 对象数据类型将在《MATLAB 外部编程接口》一书中详细讲述。

3.2　MATLAB 基本数值类型

MATLAB 的基本数值类型变量或者对象主要用来描述基本的数值对象，例如双精度数据或者整数类型的数据。在 MATLAB 还存在一类数据——常量数据，常量数据是指在使用 MATLAB 过程中由 MATLAB 提供的公共数据，这些数据可以通过数据类型转换的方法转换常量到不同的数据类型，还可以被赋予新的数值。在 MATLAB 中还有一种数据叫作空数组或者空矩阵，在创建数组或者矩阵时，可以使用空数组或者空矩阵辅助创建数组或者矩阵。本小节将详细讨论这些内容。

3.2.1　基本数值类型入门

表 3-1 中总结了 MATLAB 的基本数值类型。

表 **3-1**　**MATLAB** 的基本数值类型

数据类型	说　明	字节数	取　值　范　围
double	双精度数据类型	8	
sparse	稀疏矩阵数据类型	N/A	
single	单精度数据类型	4	
uint8	无符号 8 位整数	1	0～255
uint16	无符号 16 位整数	2	0～65 535
uint32	无符号 32 位整数	4	0～4 294 967 295
uint64	无符号 64 位整数	8	0～18 446 744 073 709 551 615
int8	有符号 8 位整数	1	- 128～127
int16	有符号 16 位整数	2	- 32 768～32 767
int32	有符号 32 位整数	4	- 2 147 483 648～2 147 483 647
int64	有符号 64 位整数	8	- 9 223 372 036 854 775 808～9 223 372 036 854 775 807

说明：

表格中所指的字节数是指使用该数据类型创建数组或者矩阵时，每一个元素占用的内存字节数，稀疏矩阵则不同，由于稀疏矩阵使用了特殊的存储数据方法，所以稀疏矩阵对象占用的内存字节数比较特殊。

复数数据类型也相对特殊，复数可以用表格中所示的各种数据类型创建，但是由于复数由实部数据和虚部数据组成，所以占用的字节数为构成复数的数据类型的两倍，例如复数 z= 1+i，在 MATLAB 中占用了 16 个字节的内存。

MATLAB 中有部分函数和这些数据类型相关，其中最常用的一个函数为 class 函数，该函数可以用来获取变量或者对象的类型，也可以用来创建用户自定义的数据类型。在本章中，主要利用其获取变量或者对象的功能。

下面结合具体的例子来说明不同的数据类型的使用方法。

例子 3-1　使用不同的数据类型。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = [ 1 2 3];
>> class(A)
ans =
double
>> whos
  Name         Size                    Bytes  Class
  A            1x3                        24  double array
  ans          1x6                        12  char array
Grand total is 9 elements using 36 bytes
>> B = int16(A);
>> class(B)
ans =
int16
>> whos
  Name         Size                    Bytes  Class
  A            1x3                        24  double array
  B            1x3                         6  int16 array
  ans          1x5                        10  char array
Grand total is 11 elements using 40 bytes
```

在例子 3-1 中，使用了 int16，即 16 位的有符号整数类型，作为示例，并且使用 class 函数获取不同变量或者对象的数据类型。向量 B 是从向量 A 通过数据类型转换得到的，可以看出，A 和 B 向量包含同样的数据，但是由于两个向量的数据类型不同，所以它们占据的内存字节数不同，其中双精度类型的向量 A 占用了 24 个字节，而 16 位整数类型的向量 B 仅占用了 6 个字节。

注意：

MATLAB 和 C 语言在处理数据类型和变量时不同。在 C 语言中，任何变量在使用之前都必须声明，然后赋值，在声明变量时就指定了变量的数据类型。但是在 MATLAB 中，任何数据变量都不需要预先的声明，MATLAB 将自动地将数据类型设置为双精度类型，若需

要使用其他类型的数据，则必须通过数据类型的转换完成。MATLAB 的数据类型名称同样就是数据类型转换的函数，利用这些函数来完成相应的数据类型转化的工作。

关于数据类型转换函数的使用可以参阅相应的在线帮助。

例子 3-2　使用不同的数据类型。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> a = [ 1 2 3];
>> b = [ 3 4 5];
>> c = a+b;
>> whos
  Name        Size                    Bytes  Class
  a           1x3                        24  double array
  b           1x3                        24  double array
  c           1x3                        24  double array
Grand total is 9 elements using 72 bytes
>> int16(a)+int16(b)
??? Error using ==> +
Function '+' is not defined for values of class 'int16'.
```

由于 MATLAB 系统默认的运算都是针对双精度类型的数据或变量的，所以在进行两个 int16 类型的变量加法时，MATLAB 报告了相应的错误。一般地，对于在表格 3-1 中所列的各种数据类型(稀疏矩阵除外)的变量， MATLAB 没有提供相应的运算规则或者计算函数。如果需要为这些没有运算规则的数据类型创建相应的计算法则，则需要利用 MATLAB 的 M 语言进行面向对象的编程，通过重载运算符来完成相应的运算定义。

有关运算符重载请读者查阅有关面向对象的书籍资料，而 M 语言的面向对象编程方面的知识请参阅 MATLAB 的帮助文档。

注意：

稀疏矩阵的元素仅能使用双精度类型的变量，sparse 类型的数据变量和整数类型数据、单精度数据类型变量之间的转换是非法的。另外，在进行数据类型转换时，若输入参数的数据类型就是需要转换的数据类型，则 MATLAB 忽略转换，保持变量的原有特性。

3.2.2　整数类型数据运算

MATLAB 运算的基本数据类型为双精度的数据类型，MATLAB 还另外提供了一些函数可以进行整数类型数据的运算，在表 3-2 中总结了这些函数。

表 **3-2**　整数类型数据的运算函数

函　数	说　　明
bitand	数据位“与”运算
bitcmp	按照指定的数据位数求数据的补码
bitor	数据位“或”操作
bitmax	最大的浮点整数数值，一般为 2^{53}- 1=9 007 199 254 740 991
bitxor	数据位“异或”操作
bitset	将指定的数据位设置为 1
bitget	获取指定的数据位数值
bitshift	数据位移操作

注意：

整数类型数据运算函数的操作数一般为无符号的整数，不同的函数会有所不同，函数运算的结果必须小于 bitmax 代表的数值。

提示：数据位运算

了解计算机中数据的二进制表达是进行二进制位运算的基础。在计算机中的任何数据都是由二进制数来保存的，计算机最初能够处理的也只有二进制的数据。对于整数可以使用原码、反码、补码来表示。

字节和位

字节这个概念对于计算机用户来讲并不陌生，计算机中的存储器就是由许许多多被称为“字节”(byte)的单元组成的。

一般地，内存的最小度量单位是位(bit)，有些地方叫作比特。而一个字节就是由 8 个二进制位组成的，其中，最右边的一位叫最低位，最左边的一位叫作最高位。所以在本章前面小节介绍的 16 位整数占用两个字节的内存，依次类推，32 位整数就需要占用四个字节的内存。

在计算机中表示数据可以有不同的方法，一般有原码、反码和补码三种形式。

为了便于表述，这里所有的数字都将按照数字在八位计算机内的表示方式来表示。

原码

将最高位作为符号位(以数字 0 表示正，以数字 1 表示负)，其他的数字位代表数值本身的绝对值，这种表示数字的方式叫作原码。

例如，数字 7 在八位计算机中的原码为 0000 0111；而数字-7 在八位计算机中的原码为 1000 0111。

如果这两个数字在我们日常使用的 32 位计算机中用原码表示，则无非再多几个数字 0。例如在 32 位的计算机中用原码表示数字 7，则为 0000 0000 0000 0000 0000 0000 0000 0111。

反码

用反码表示数字的规则是，如果是正数，则用这个数字的原码来表示，如果是负数，则保持符号位为 1，然后将这个数字的原码按照每位取反，得到这个数字的反码。

例如，数字 7 在八位计算机中的反码就是它的原码，为 000 0111；而数字- 7 在八位计算机中的反码是 1111 1000。

补码

计算机中表示数据的一般方式是补码，补码是这样规定的：

一个正数的补码就是其原码，例如整数 7 在八位计算机中的补码是 0000 0111。

对于负数的补码是将数字的反码加上 1，就得到了这个数字的补码。例如-7 在 8 位计算机中的补码是 1111 1001。

关于数字在计算机中二进制表示的详细解释请参阅相应的计算机原理方面的书籍。

MATLAB 整数类型数据运算和 C 语言整数位运算比较类似，所不同的是 MATLAB 中没有相应的运算符，而只有这些函数可用，并且这些整数运算的数据都必须大于 0。这里结合具体的示例讲解表 3-2 所列函数的用法。

例子 3-3(a)　数据位“与”操作。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = 86; B = 77;
>> C = bitand(A,B)
C =
     68
>> a = uint16(A);b = uint16(B);
>> c = bitand(a,b)
c =
     68
>> whos
  Name        Size                      Bytes  Class
  A           1x1                           8  double array
  B           1x1                           8  double array
  C           1x1                           8  double array
  a           1x1                           2  uint16 array
  b           1x1                           2  uint16 array
  c           1x1                           2  uint16 array
Grand total is 6 elements using 30 bytes
```

请读者注意例子 3-3(a)运算得到结果的数据类型，数据 a、b 为无符号的 16 位整数，所以进行操作的结果也是无符号的 16 位整数，但数据 A、B 为双精度类型的数据(注意这些数据都是非负整数)，结果运算的结果是双精度类型。

运算的过程大体如下：

86 的补码	0101 0110	
77 的补码	0100 1101	
“与”运算的结果	0100 0100	(68)

例子 3-3(b)　整数数据位的运算。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = 86;
>> dec2bin(A)
ans =
1010110
>> B = bitset(A,6);
>> dec2bin(B)
ans =
1110110
>> C = bitset(A,7,0);
>> dec2bin(C)
ans =
10110
>> D = bitshift(A,4);
>> dec2bin(D)
ans =
```

```
10101100000
>> E = bitshift(A, - 4);
>> dec2bin(E)
ans =
101
>> a = uint16(A);
>> e = bitshift(A, - 4);
>> dec2bin(e)
ans =
101
>> whos
  Name      Size                    Bytes  Class
  A         1x1                         8  double array
  B         1x1                         8  double array
  C         1x1                         8  double array
  D         1x1                         8  double array
  E         1x1                         8  double array
  a         1x1                         2  uint16 array
  ans       1x3                         6  char array
  e         1x1                         8  double array
Grand total is 10 elements using 56 bytes
```

说明：

dec2bin 函数是将十进制整数转变成二进制整数字符串的函数，该函数将在本章 3.4.4 字符串转换函数一节详细讲述。

在例子 3-3(b)中，使用了 bitset 函数和 bitshift 函数，其中 bitshift 函数类似于 C 语言的“>>”运算符和“<<”运算符，如果函数输入的第二个参数为正数，则进行左位移操作，否则为右位移操作。bitset 函数根据输入的第二个参数设置相应的数据位的数值，若不指定第三个参数，则将相应的数据位设置为“1”，否则根据输入的第三个参数(0 或者 1)设置相应的数据位。另外，注意一些函数运算的结果，有些函数的运算结果是双精度数据类型。

3.2.3 MATLAB 的常量

表 3-3 中总结了在 MATLAB 中预定义的常量。

表 **3-3**　**MATLAB** 的常量

常　量	说　　明
ans	最近运算的结果
eps	浮点数相对精度
realmax	MATLAB 能够表示的实数的最大绝对值
realmin	MATLAB 能够表示的实数的最小绝对值
pi	常数 p
i,j	复数的虚部数据最小单位
Inf	无穷大
NaN	非数(Not a Number)

说明：

eps、realmax 和 realmin 三个常量具体的数值与运行 MATLAB 的计算机相关，不同的计算机系统可能具有不同的数值，例如，在笔者的计算机上，这三个数值分别为 eps=2.2204 $\times 10^{-16}$， realmax=1.7977$\times 10^{308}$，realmin=2.2251$\times 10^{-308}$。

和其他的高级编程语言所定义的常量不同，MATLAB 的常量数值是可以修改的。例如，在 MATLAB 命令行窗口中可以键入如下的指令：pi=100，这样 pi 这个常量的数值就变成了 100。但是，如果用 clear 指令清除变量后，pi 将变成原有的常量数值。

Inf 也可以写作 inf，它为 IEEE 定义的算术数据无穷大数值，在 MATLAB 中进行诸如 1.0/0.0 或者 exp(1000)的操作时都会得到这个数值。如果将 inf 应用于函数，则计算结果可能为 inf 或者 NaN。

NaN 也可以写作 nan，它为 IEEE 规定的某种运算得到的结果，例如 0/0 的运算得到的结果就是 NaN。NaN 参与运算的结果也为 NaN(关系运算除外)。

例子 3-4　**NaN** 和 **Inf** 运算示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> a = inf;
>> class(a)
ans =
double
>> b = int16(a)
b =
       32767
>> c = sin(a)
c =
       NaN
>> sin(c)
ans =
       NaN
>> class(c)
ans =
double
>> int64(c)
ans =
         0
```

说明：

MATLAB 中所有的数据默认类型均为双精度类型，包括像 NaN 和 Inf 在内的上述若干常数。

对 NaN 和 Inf 进行数据转化时要注意，Inf 将获取相应数据类型的最大值，而 NaN 往往返回相应整数数据类型的数值 0，浮点数类型则仍然为 NaN。

在运算中使用 NaN 可以避免因为执行了 0/0 这类能够产生错误的应用程序中断，可以

辅助调试应用程序。

例子 3-5　最小复数单位的使用。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> a = i
a =
      0 + 1.0000i
>> i = 1
i =
      1
>> b = i+j
b =
   1.0000 + 1.0000i
>> clear
>> c = i+j
c =
           0 + 2.0000i
```

通过例子 3-5 可以看出，MATLAB 的常量是可以赋予新的数值的。一旦被赋予了新的数值，则常量代表的就不是原有的数值，而是新的数值，除非执行 clear 命令。就像创建变量 c 的时候，由于执行了 clear 指令，将工作空间中的变量清除干净，于是在执行 c = i+j 的时候，MATLAB 认为这是在进行两个最小复数运算单位的计算，所以得到了 c = 2i。

3.2.4　空数组

所谓空数组，就是指那些某一个维或者某些维的长度为 0 的数组。它是为了完成某些 MATLAB 操作和运算而专门设计的一种数组。利用空数组可以修改数组的大小，但是不能修改数组的维数。

下面通过具体的例子来说明空数组创建和使用的过程。

例子 **3-6**　创建空数组。

和创建普通的数组(矩阵)一样，创建空数组也有不同的方法，在 MATLAB 命令行窗口中键入下面的命令：

```
>> A = []
A =
      []
>> B = ones(2,3,0)
B =
   Empty array: 2-by-3-by-0
>> C = randn(2,3,4,0)
C =
   Empty array: 2-by-3-by-4-by-0
```

```
>> whos
  Name        Size                     Bytes  Class
  A           0x0                          0  double array
  B           2x3x0                        0  double array
  C           4-D                          0  double array
Grand total is 0 elements using 0 bytes
```

空数组并不意味着什么都没有，使用 whos 命令可以看到空数组类型的变量在 MATLAB 的工作空间中确实存在。在数组编辑器中也可以对空数组进行编辑，填充矩阵的元素。图 3-2 为空数组在数组编辑器中显示的状况。

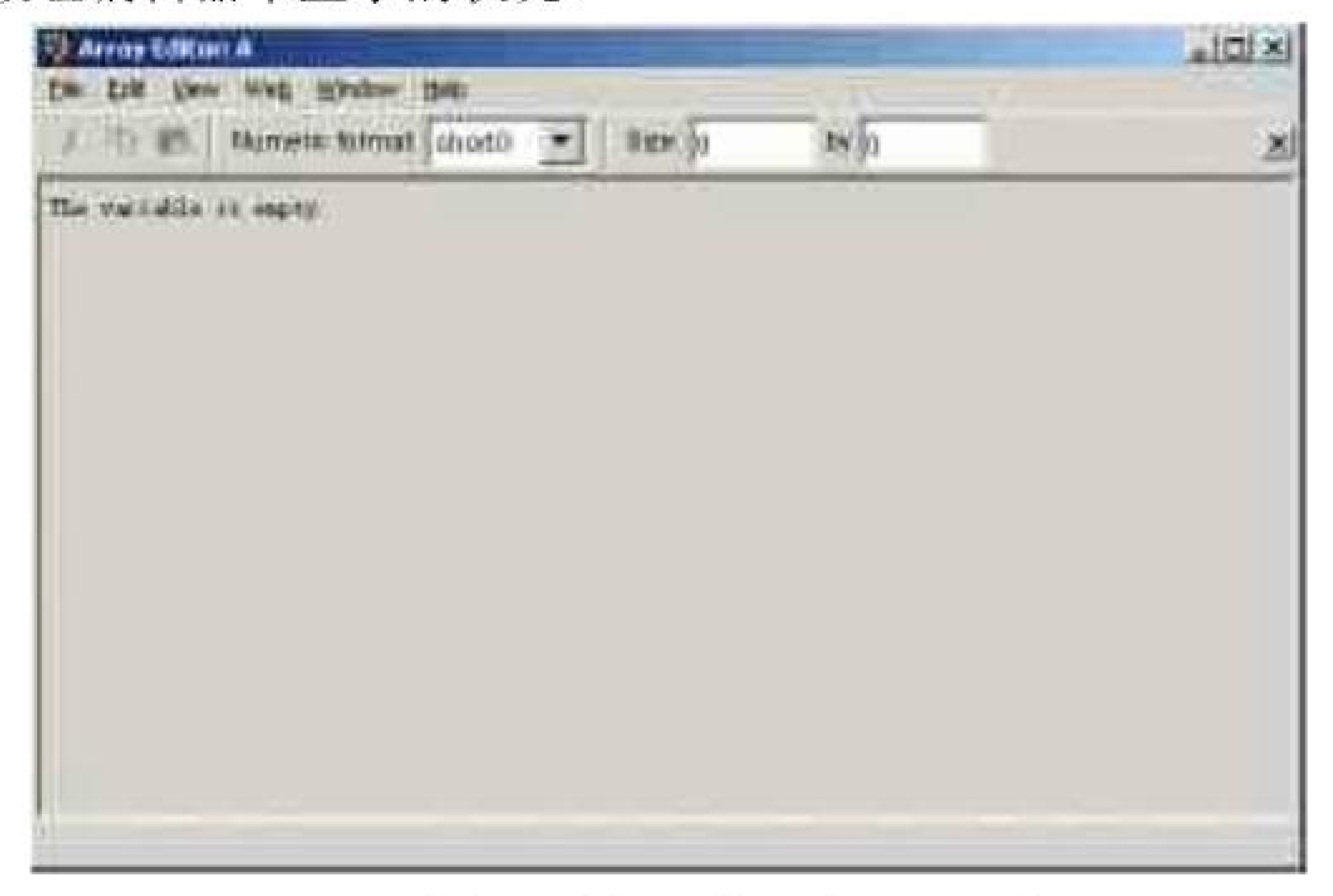


图 3-2　空数组在数组编辑器中的显示情况

使用空数组可以完成一些比较特殊的操作，例如使用空数组可以将大数组删除部分行或者列，也可以删除多维数组的某一页，见下面的例子。

例子 3-7　使用空数组的例子。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A (2,2,3) = 1
A(:,:,1) =
     0     0
     0     0
A(:,:,2) =
     0     0
     0     0
A(:,:,3) =
     0     0
     0     1
>> A(:,:,3) = []
A(:,:,1) =
     0     0
     0     0
```

```
A(:,:,2) =
     0     0
     0     0
>> B = reshape(1:24,4,6)
B =
     1     5     9    13    17    21
     2     6    10    14    18    22
     3     7    11    15    19    23
     4     8    12    16    20    24
>> B(:,[2 3 4]) = []
B =
     1    17    21
     2    18    22
     3    19    23
     4    20    24
```

从例子 3-7 可以看出，利用空数组可以非常方便地将数组或者矩阵的部分元素删除。例如例子中将矩阵 B 的第二、三、四列删除得到了新的数组(矩阵)，而对于多维数组 A 的操作则删除了数组的第三页，使其成为了具有二页的三维数组。

3.3　逻辑类型和关系运算

在大多数的高级编程语言中都有所谓的逻辑数据类型，用来完成诸如关系运算或者逻辑运算。虽然在标准的 C 语言中没有逻辑数据类型，但是仍然定义非零值为逻辑真，零值为逻辑假。在 MATLAB 中也有相应的操作和数据类型，分别叫作逻辑运算和逻辑数据类型，同样，在 MATLAB 中也有相应的关系运算。

3.3.1　逻辑数据类型

所谓逻辑数据类型就是仅具有两个数值的一种数据类型，其中，一个数值为 TRUE，另外一个数值为 FALSE。在 MATLAB 中，参与逻辑运算或者关系运算的并不一定必须有逻辑类型的数据，任何数值都可以参与逻辑运算。这时，MATLAB 将所有非零值看作逻辑真，将零值看作逻辑假。一般地，1 表示逻辑真，0 表示逻辑假。

和一般的数值类型类似，逻辑类型的数据只能通过数值类型转换，或者使用特殊的函数生成相应类型的数组或者矩阵。

创建逻辑类型矩阵或者数组的函数主要有三个，见表 3-4。

表 **3-4**　创建逻辑类型数据的函数

函 数	说　明
logical	将任意类型的数组转变成为逻辑类型数组，其中，非零元素为真，零元素为假
True	产生逻辑真值数组
False	产生逻辑假值数组

使用表 3-4 中的函数都可以创建相应的逻辑类型数组，用法参见例子 3-8。

例子 3-8　创建逻辑类型数组。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = eye(3);
>> B = logical(A)
B =
     1     0     0
     0     1     0
     0     0     1
>> C = true(size(A))
C =
     1     1     1
     1     1     1
     1     1     1
>> D = false([size(A),2])
D(:,:,1) =
     0     0     0
     0     0     0
     0     0     0
D(:,:,2) =
     0     0     0
     0     0     0
     0     0     0
>> whos
  Name      Size                    Bytes  Class
  A         3x3                        72  double array
  B         3x3                         9  logical array
  C         3x3                         9  logical array
  D         3x3x2                      18  logical array
Grand total is 45 elements using 108 bytes
```

使用 logical 函数、true 函数和 false 函数的过程都比较简单，通过最后的比较可以看出，逻辑类型的数组每一个元素仅占用一个字节的内存空间，所以矩阵 B 尽管和矩阵 A 看上去一致，但是内存占用上有相当大的差距，并且属于不同的数据类型，也就有不同的操作函数和方法。

注意：

本书将 MATLAB 的 logical array(逻辑数组)称为逻辑类型的数组。在有些书籍上，将 MATALB 的这种数据类型直接叫做布尔类型数组，请读者注意对比。

逻辑类型的数组元素仅能有两个取值：1 或者 0，分别表示逻辑真和逻辑假。

另外，在 MATLAB 中还有若干函数以 is 开头，这类函数是用来完成某种判断功能的函

数，例如函数 isnan 判断输入参数是否为 NaN，而 isnumeric 函数判断输入参数是否为某种数值类型，见例子 3-9。

例子 3-9　isnumeric 函数的使用示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> a =true
a =
      1
>> isnumeric(a)
ans =
      0
>> b = 1;
>> isnumeric(b)
ans =
      1
```

注意：

在使用 true 或者 false 函数创建逻辑类型数组时，若不指明参数，则创建一个逻辑类型的标量。

此外，能够产生逻辑数据类型结果的运算还有关系运算，关系运算将在 3.3.3 小节详细讨论。

3.3.2　逻辑运算

能够处理逻辑类型数据的运算叫作逻辑运算，在 MATLAB 能够处理的逻辑类型运算和 C 语言比较类似，见表 3-5。

表 3-5　**MATLAB** 的逻辑运算

运 算 符	说　　明
&&	具有短路作用的逻辑与操作，仅能处理标量
||	具有短路作用的逻辑或操作，仅能处理标量
&	元素与操作
|	元素或操作
~	逻辑非操作
xor	逻辑异或操作
any	当向量中的元素有非零元素时，返回真
all	当向量中的元素都是非零元素时，返回真

说明：

参与逻辑运算的操作数不一定必须是逻辑类型的变量或常数，也可以使用其他类型的数据进行逻辑运算，但是运算的结果一定是逻辑类型的数据。

所谓具有短路作用是指，在进行&&或||运算时，若参与运算的变量有多个，例如 a && b && c && d，若 a、b、c、d 四个变量中 a 为假，则后面的三个都不再被处理，运算结束，并返回运算结果逻辑假。

例子 3-10　逻辑运算示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> a = eye(3);
>> b = a;b(3,1) =1;
>> a && b
??? Operands to the || and && operators must be convertible to logical scalar values.
>> a & b
ans =
      1      0      0
      0      1      0
      0      0      1
>> whos
  Name        Size                    Bytes  Class
  a           3x3                        72  double array
  ans         3x3                         9  logical array
  b           3x3                        72  double array
Grand total is 27 elements using 153 bytes
```

例子 3-10 中，首先参与逻辑运算的是两个双精度类型的矩阵，这两个矩阵进行&&运算时，MATLAB 报告了相应的错误，因为参与&&或者||运算的操作数必须为标量。另外，从例子也能够看出，尽管参与运算的是两个双精度类型的矩阵，运算的结果却是逻辑类型的矩阵。

例子 3-11　函数 all 和 any 使用示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> a = [1 0 1;1 0 0 ;1 1 0;1 1 1]
a =
      1      0      1
      1      0      0
      1      1      0
      1      1      1
>> all(a)
ans =
      1      0      0
>> any(a)
ans =
      1      1      1
```

在例子 3-11 中，函数 all 和函数 any 可以针对矩阵中每一列进行处理。函数 all 的作用是若每列元素均为非零值，则返回逻辑真，函数 any 的作用是若每列元素有非零值，则返回逻辑真。

3.3.3　关系运算

关系运算是用来判断两个操作数关系的运算，MATLAB 中的关系运算和 C 语言的关系运算基本一致，主要有六种，见表 3-6。

表 **3-6**　**MATLAB** 中的关系运算符

运算符	说明	运算符	说明
= =	等于	>	大于
~=	不等于	<=	小于等于
<	小于	>=	大于等于

参与关系运算的操作数可以使用各种数据类型的变量或者常数，运算的结果是逻辑类型的数据。标量也可以和矩阵或者数组进行比较，比较的时候将自动扩展标量，返回的结果是和数组同维的逻辑类型数组。如果进行比较的是两个数组，则数组必须是同维的，且每一维的尺寸也必须一致。

例子 3-12　关系运算示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = reshape(1:9,3,3);
>> B = magic(3);
>> A > B
ans =
     0     1     1
     0     0     1
     0     0     1
>> A = = B
ans =
     0     0     0
     0     1     0
     0     0     0
```

结合前面小节讲述的逻辑运算和本小节的关系运算可以完成更复杂的关系运算和处理。如前文所述，参与逻辑运算的操作数是逻辑类型的数据，所以，利用“( )”和各种运算符相结合，可以完成复杂的关系运算，见例子 3-13。

例子 3-13　复杂的关系运算。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = reshape(-4:4,3,3)
A =
    - 4    - 1     2
    - 3      0     3
    - 2      1     4
>> B = ~(A>=0)
```

```
B =
        1     1     0
        1     0     0
        1     0     0
>> C = (A>0)&(A<3)
C =
        0     0     1
        0     0     0
        0     1     0
```

在例子 3-13 中，使用逻辑运算和关系运算从矩阵 A 中分别标识出不大于 0 的数据和大于 0 且小于 3 的数据索引位置。

将逻辑类型的数据应用于索引就构成了逻辑索引，利用逻辑索引，可以方便地从矩阵或者数组中找到某些符合条件的元素，见例子 3-14。

例子 3-14　逻辑索引示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = [-2 10 NaN 30 0 -11 -Inf 31];
>> pos = A<0
pos =
        1     0     0     0     0     1     1     0
>> B = A(pos)
B =
      - 2    - 11    - Inf
>> pos = (A>=0)&(isfinite(A))
pos =
        0     1     0     1     1     0     0     1
>> C = A(pos)
C =
      10     30     0     31
```

在例子 3-13 中，首先从向量 A 中获取了全部小于 0 的元素构成向量 B，然后从向量 A 中获取了不小于 0 且有穷的向量元素。可以看出，逻辑索引数组可以非常方便地获取数组中满足条件的元素，这种操作在处理大数组时非常有效。

3.3.4　运算符的优先级

到本小节为止，MATLAB 的基本运算符都已经介绍完毕了。在 M 语言中可以将这些不同的运算符组合起来创建复杂的运算表达式。M 语言的运算符和普通的高级编程语言类似，也具有相应的计算优先级。这里将 M 语言的运算以及相应的计算优先级进行了总结：

(1) 括号( )。

(2) 数组转置(.')，数组幂(.^)，复转置(')，矩阵幂(^)。

(3) 一元加(+)，一元减(-)，逻辑非(~)。

(4) 数组乘法(.*)，数组除法(./)，数组左除(.\)，矩阵乘法(*)，矩阵右除(/)，矩阵左除(\)。

(5) 加法(+)，减法(-)。

(6) 冒号运算符(:)。

(7) 小于(<)，小于等于(<=)，大于(>)，大于等于(>=)，等于(= =)，不等于(~=)。

(8) 元素与(&)。

(9) 元素或(|)。

(10) 短路逻辑与(&&)。

(11) 短路逻辑或(||)。

上面的运算符优先级是由高到低排列的，例如括号运算符的优先级最高，数组转置等次之。如果同一级别的运算符出现在表达式中，则按照运算符在表达式中出现的次序，由左向右排列。在使用 M 语言编写程序时，需要灵活使用这些运算符来具体实现不同的算法。

3.4 字 符 串

MATLAB 的字符串，部分书籍称字符串为字符串数组或者字符数组，它是一种相对比较次要的数据类型，但是这种数据类型也不可缺少。在数据的可视化、应用程序的交互方面，字符串都起到非常重要的作用。创建字符串时需要使用单引号将字符串的内容包含起来，字符串一般以行向量形式存在，并且每一个字符占用两个字节的内存。

注意：

可以认为 MATLAB 中没有字符类型的变量，所有的字符都是以字符串的形式存在。这一点和一些高级语言不同。

3.4.1 字符串入门

在详细了解字符串之前，首先了解一下创建字符串的方法，例子 3-15 说明了创建字符串的方法，以及字符串和其他数值类型的区别。

例子 3-15　字符串的创建。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> a = 127
a =
    127
>> class(a)
ans =
double
>> size(a)
ans =
       1       1
>> b = '127'
```

```
b =
      127
>> class(b)
ans =
      char
>> size(b)
ans =
       1      3
>> whos
   Name          Size                    Bytes  Class
   a             1x1                         8  double array
   ans           1x2                        16  double array
   b             1x3                         6  char array
Grand total is 6 elements using 30 bytes
```

创建字符串时，只要将字符串的内容用单引号包含起来就可以了，若需要在字符串内容中包含单引号，则需要在键入字符串内容时，连续键入两个单引号即可，例如

```
>> c = 'Isn''t it?'
c =
Isn't it?
```

另外，使用 char 函数可以创建一些无法通过键盘输入的字符，该函数的作用是将输入的整数参数转变成为相应的字符，一般地转换成为相应的 Unicode 字符。同样字符串也可以转变成为相应的双精度数据。关于 char 函数的使用将在后续章节中讲述。

关于 **Unicode**

设计 Unicode 的主要目的是为了简化对非罗马语系字符的处理。依照 Unicode 设计的宗旨，Unicode 将对世界上所有语言的字符进行控制，所以，Unicode 占用了两个字节，也就是说 Unicode 可以表示 65 536 个字符。同用单个字节表示字符的 ASCII/ANSI 字符相比，Unicode 字符集的容量大了很多，而且，ASCII/ANSI 字符集是 Unicode 的子集，Unicode 字符集的前 256 个字符就是 ASCII 字符。

目前已经有阿拉伯文、中文拼音、俄文、希腊文等多种语言定义了 Unicode 代码点，总计约有 35 000 个，还有约 29 000 个代码点等待分配，此外，还有约 6000 个代码点供个人使用。

开发应用程序时考虑使用 Unicode 的主要原因是：

? 可以比较容易地在不同语言之间交换数据。

? 可以使自己编写的二进制文件支持所有的语言版本。

? 提高应用程序的运行效率。

目前，在 Windows 2000 和 Windows XP 等操作系统中，所有的核心函数都要求使用 Unicode 字符。

欲了解 Unicode 的详细情况，请登录 www.unicode.org 网站。

3.4.2　基本字符串操作

本小节详细介绍字符串的基本操作，结合具体的例子将详细讲解字符串元素索引、字符串的拼接、字符串与数值之间的转换等操作。

例子 3-16　字符串元素索引。

字符串实际上也是一种 MATLAB 的向量或者数组，所以，一般利用索引操作数组的方法都可以用来操作字符串。在 MATLAB 命令行窗口中，键入下面的指令：

```
>> a = 'This is No.3-15 Example!'
a =
This is No.3-15 Example!
>> b = a(1:4)
b =
This
>> c = a(12:15)
c =
3-15
>> d = a(17:end)
d =
Example!
```

本例子使用了索引获取字符串 a 的子串，直观上在字符串中使用索引和在数组或向量中使用索引是没有任何区别的。

字符串还可以利用“[]”运算符进行拼接，不过拼接字符串时需要注意以下两点：

? 若使用“，”作为不同字符串之间的间隔，则相当于扩展字符串成为更长的字符串向量。

? 若使用“；”作为不同字符串之间的间隔，则相当于扩展字符串成为二维或者多维的数组，这时，不同行上的字符串必须具有同样的长度，见例子 3-17。

例子 3-17　字符串的拼接。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> a = 'Hello';
>> b = 'MOTO!';
>> length(a) == length(b)
ans =
     1
>> c = [a, ' ', b]
c =
Hello MOTO!
>> d = [ a ; b]
```

```
d =
     Hello
     MOTO!
>> size(c)
ans =
      1      11
>> size(d)
ans =
      2      5
```

在例子 3-16 中，首先创建了两个长度一致的字符串，然后利用“[]”运算符将两个字符串分别组合成为了向量 c 和矩阵 d。不过在创建矩阵 d 时，各个行字符串必须具有相同的长度。

拼接字符串还可以使用部分函数完成，这些函数在 3.4.3 小节讲述。

如前文所述，在 MATLAB 中使用了 Unicode 作为字符集，所以每一个字符占用了两个字节的存储空间。字符串和一般的数值之间也可以进行相应的转换，在例子 3-18 中，将使用字符向数值转换的方法察看相应字符的 Unicode 数值。

例子 3-18　字符串和数值的转换。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> a = 'Hello MOTO!';
>> b = double(a)
b =
  Columns 1 through 8
    72    101    108    108    111     32     77     79
  Columns 9 through 11
    84     79     33
>> c = '您好！';
>> d = double(c)
d =
        50426          47811          41889
>> char(d)
ans =
您好！
```

前文曾经提及，使用 char 函数能够将数值转变成为字符，那么将字符转变成数值则必须使用 double 函数。

3.4.3　字符串操作函数

MATLAB 提供了一些函数用来操作字符串，在表 3-7 中进行了总结。

表 **3-7**　常用的字符串操作函数

函　数	说　　明
char	创建字符串，将数值转变成为字符串
double	将字符串转变成 Unicode 数值
blanks	创建空白的字符串(由空格组成)
deblank	将字符串尾部的空格删除
ischar	判断变量是否是字符类型
strcat	水平组合字符串，构成更长的字符向量
strvcat	垂直组合字符串，构成字符串矩阵
strcmp	比较字符串，判断字符串是否一致
strncmp	比较字符串前 n 个字符，判断是否一致
strcmpi	比较字符串，比较时忽略字符的大小写
strncmpi	比较字符串前 n 个字符，比较时忽略字符的大小写
findstr	在较长的字符串中查寻较短的字符串出现的索引
strfind	在第一个字符串中查寻第二个字符串出现的索引
strjust	对齐排列字符串
strrep	替换字符串中的子串
strmatch	查询匹配的字符串
upper	将字符串的字符都转变成为大写字符
lower	将字符串的字符都转变成为小写字符

下面结合具体的示例讲解部分函数的具体用法。

例子 3-19　组合字符串示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> a = 'Hello';
>> b = 'MOTO!';
>> c = strcat(a,b)
c =
HelloMOTO!
>> d   =strvcat(a,b,c)
d =
Hello
MOTO!
HelloMOTO!
>> whos
  Name        Size                   Bytes  Class
  a           1x5                       10  char array
  b           1x5                       10  char array
  c           1x10                      20  char array
  d           3x10                      60  char array
Grand total is 50 elements using 100 bytes
```

在例子 3-19 中，使用 strcat 函数和 strvcat 函数进行了字符串的组合。其中，与前一小节介绍的字符串组合不同，strvcat 函数允许将不同长度的字符串组合成为字符矩阵，并且将短字符串扩充为与长字符串相同的长度。

例子 3-20　字符串比较函数应用示例。

在 MATLAB 命令行窗口中，键入如下的指令：

```
>> a = 'The first string';
>> b = 'The second string';
>> c = strcmp(a,b)
c =
     0
>> d = strncmp(a,b,4)
d =
     1
>> whos
  Name       Size                    Bytes  Class
  a          1x16                       32  char array
  b          1x17                       34  char array
  c          1x1                         1  logical array
  d          1x1                         1  logical array
Grand total is 35 elements using 68 bytes
```

在例子 3-20 中，使用两种不同函数进行了字符串比较，strcmp 比较两个字符串的全部字符，所以第一次比较时，系统返回了逻辑假值，而 strncmp 只比较指定字符串中的前 n 个字符，所以在第二次比较时，系统返回了逻辑真值。

另外，使用 isequal 函数也可以比较两个或两个以上字符串是否一致。

例子 3-21　findstr 函数和 strfind 函数使用示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> S1 = 'A friend in need is a friend indeed';
>> S2 = 'friend';
>> a = findstr(S2,S1)
a =
     3    23
>> b = strfind(S2,S1)
b =
     []
>> whos
  Name       Size                    Bytes  Class
  S1         1x35                       70  char array
  S2         1x6                        12  char array
  a          1x2                        16  double array
```

```
b            0x0                    0  double array
Grand total is 43 elements using 98 bytes
```

在例子 3-21 中，分别使用 findstr 函数和 strfind 函数在字符串中查寻子串的索引位置。注意两个函数的区别，findstr 函数从长字符串中查寻短字符串的索引位置，而 strfind 是在第一个字符串参数中查寻第二个字符串参数的索引。所以，在例子 3-21 中两次操作获取了不同的运算结果。

3.4.4 字符串转换函数

在 MATLAB 中允许不同类型的数据和字符串类型的数据之间进行转换，这种转换需要使用不同的函数完成。另外，同样的数据，特别是整数数据有很多种表示的格式，例如十进制、二进制或者十六进制。在 C 语言中，printf 函数通过相应的格式字符串就可以输出不同格式的数据。而在 MATLAB 中，则直接提供了相应的函数完成数制的转换。在表 3-8 和表 3-9 中，分别列举了这些函数。

表 **3-8** 数字和字符之间的转换函数

函 数	说 明
num2str	将数字转变成为字符串
int2str	将整数转变成为字符串
mat2str	将矩阵转变成为可被 eval 函数使用的字符串
str2double	将字符串转变为双精度类型的数据
str2num	将字符串转变为数字
sprintf	格式化输出数据到命令行窗口
sscanf	读取格式化字符串

表 **3-9** 不同数值之间的转换函数

函 数	说 明
hex2num	将十六进制整数字符串转变成为双精度数据
hex2dec	将十六进制整数字符串转变成为十进制整数
dec2hex	将十进制整数转变成为十六进制整数字符串
bin2dec	将二进制整数字符串转变成为十进制整数
dec2bin	将十进制整数转变成为二进制整数字符串
base2dec	将指定数制类型的数字字符串转变成为十进制整数
dec2base	将十进制整数转变成为指定数制类型的数字字符串

在表 3-8 中列举的数字与字符串之间转换的函数中，较为常用的函数是 num2str 和 str2num，这两个函数将在讲述 MATLAB 图形用户界面(GUI)编程中大量用到，例子 3-22 中讨论了这两个函数的用法。

例子 3-22 num2str 函数和 str2num 函数的用法示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> S = ['1 2';'2 3'];
>> A = str2num(S)
A =
```

```
    1       2
    2       3
>> B = str2num('1 + 2i')
B =
   1.0000 + 2.0000i
>> C = str2num('1 +2i')
C =
   1.0000                    0 + 2.0000i
>> D = num2str(rand(2,2),5)
D =
0.95013      0.60684
0.23114      0.48598
>> whos
  Name       Size                 Bytes  Class
  A          2x2                     32  double array
  B          1x1                     16  double array (complex)
  C          1x2                     32  double array (complex)
  D          2x19                    76  char array
  S          2x3                     12  char array
Grand total is 51 elements using 168 bytes
```

在使用 str2num 函数时需要注意如下几点：

? 被转换的字符串仅能包含数字、小数点、字符“e”或者“d”、数字的正号或者负号，以及复数虚部字符“i”或者“j”，此外不可以包含其他字符。

? 使用该字符转换函数时需要注意空格。例如在本例子中转换生成变量 B 和 C 时得到了不同的结果，主要原因是在变量 C 中，字符“1”和字符“+2i”之间存在空格，而加号“+”和数字 2 之间没有空格，所以转化的结果与生成变量 B 时不同，创建变量 B 的时候，在数字 1、加号“+”和数字 2 之间都存在空格。

? 为了避免上述问题，可以使用 str2double 函数，但是该函数仅能转换标量，不能转换矩阵或者数组。

使用 num2str 将数字转换成为字符串时，可以指定字符串所表示的有效数字位数，详细信息可以参阅 MATLAB 的帮助文档或者 MATLAB 在线帮助。

例子 3-23　数制转换函数示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> a = 255;
>> h = dec2hex(a)
h =
FF
>> b = dec2bin(a)
b =
11111111
```

```
>> c = dec2base(a,5)
c =
2010
>> b(end) = '0';
>> bin2dec(b)
ans =
   254
>> whos
  Name      Size                    Bytes  Class
  a         1x1                         8  double array
  ans       1x1                         8  double array
  b         1x8                        16  char array
  c         1x4                         8  char array
  h         1x2                         4  char array
Grand total is 16 elements using 44 bytes
```

在例子 3-23 中，使用了部分数制转换的函数，其中比较特殊的一个是 dec2base 函数，它和 base2dec 函数类似，这两个函数的输入参数为两个，第二个参数表示相应的数制，比如在例子 3-23 中调用的函数，是将十进制数据 a 转变成为了五进制数据的字符串。

注意

需要注意各种数制转换函数的输入参数类型和输出参数类型，请仔细阅读 MATLAB 的帮助文档和本书中的例子。

3.4.5 格式化输入输出

和 C 语言一致，MATLAB 也能够进行格式化的输入、输出，这是一种高级编程语言所必备的一种能力。MATLAB 继承了标准 C 语言中用于 printf 函数的格式化字符，也就是说，用于 C 语言的格式化字符串都可以用于 MATLAB 的格式化输出函数。在本小节将详细讲述 MATLAB 的格式化输入、输出函数的使用方法。表 3-10 中总结了可以应用在格式化输出函数的格式化字符。

表 **3-10**　格式化字符

字　符	说　明
%c	显示内容为单一的字符
%d	有符号的整数
%e	科学技术法，使用小写的字符 e
%E	科学技术法，使用大写的字符 E
%f	浮点数据
%g	不定，在%e 或者%f 之间选择一种形式
%G	不定，在%E 或者%f 之间选择一种形式
%o	八进制表示
%s	字符串
%u	无符号的整数
%x	十六进制表示，使用小写的字符
%X	十六进制表示，使用大写的字符

除了这些格式化字符外，MATLAB 还支持那些在 C 语言中就包含的特殊字符，例如回车符、制表符等，这里就不再赘述。

在 MATLAB 中，主要有两个函数用来进行格式化的输入和输出，这两个函数分别为 sscanf 和 sprintf。

sscanf 函数用来从字符串中获取数据，它的基本语法结构为

A = sscanf(s,format)

A = sscanf(s,format,size)

在 sscanf 函数的参数中，s 为包含数据的字符串，format 是转换字符串数据的格式化字符串，而 size 是需要转换的字符矩阵的大小。函数的使用方法参见下面的例子。

例子 3-24　sscanf 函数的例子。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> S1 = '2.7183    3.1416';
>> S2 = '2.7183e3 3.1416e3';
>> S3 = '0 2 4 8 16 32 64 128';
>> A = sscanf(S1,'%f')
A =
    2.7183
    3.1416
>> B = sscanf(S2,'%e')
B =
  1.0e+003 *
    2.7183
    3.1416
>> C = sscanf(S3,'%d')
C =
     0
     2
     4
     8
    16
    32
    64
   128
>> whos
  Name      Size                    Bytes  Class
  A         2x1                        16  double array
  B         2x1                        16  double array
  C         8x1                        64  double array
  S1        1x14                       28  char array
```

```
  S2         1x17                    34  char array
  S3         1x20                    40  char array
Grand total is 63 elements using 198 bytes
```

在使用 sscanf 函数进行格式化的输入时，需要注意输入数据格式与格式化字符串之间的匹配，否则得到的结果可能不正确。

格式化的输出函数是 sprintf 函数，该函数的基本语法如下：

s = sprintf(format , A , ………)

其中，format 是格式化字符串，A 为输出的数据，而 s 则是函数格式化得到的输出结果，熟悉 C 语言的读者理解 sprintf 函数应该没有任何问题。

例子 3-25　**sprintf** 函数的例子。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = 1/eps; B = -eps;
>> C = [ 65,66,67,pi];
>> D = [pi,65,66,67];
>> s1 = sprintf('%+15.5f',A)
s1 =
+4503599627370496.00000
>> s2 = sprintf('%+.5e',B)
s2 =
-2.22045e-016
>> s3 = sprintf('%s %f',C)
s3 =
ABC 3.141593
>> s4 = sprintf('%s %f %s',D)
s4 =
3.141593e+000 65.000000 BC
>> whos
  Name       Size                    Bytes  Class
  A          1x1                         8  double array
  B          1x1                         8  double array
  C          1x4                        32  double array
  D          1x4                        32  double array
  s1         1x23                       46  char array
  s2         1x13                       26  char array
  s3         1x12                       24  char array
  s4         1x26                       52  char array
Grand total is 84 elements using 228 bytes
```

从例子 3-24 中可以看出，利用格式化字符串输出数据的时候有如下几点需要注意：

? 格式化字符串若包含了“+”，则表示在输出的字符串中包含数据的符号，如输出字

符串 s1 和 s2 时的结果。

? 对于整数数值进行格式化输出时，可以直接将向量转变成为字符串，例如字符串 s2 的输出结果。

? 如果输出的数据与相应的格式化字符串不匹配，则输出数值最常见的一种形式，如字符串 s4 的输出结果。

作为编程语言，必须能够和用户的输入进行交互，所以 MATLAB 也提供了相应的函数用来完成获取用户输入数据的功能，这个函数就是 input 函数，它的基本用法如下：

A = input(prompt);

A = input(prompt, 's')

其中，prompt 为提示用的字符串，若具有第二个参数 s，则输入数据为字符串，否则为双精度数据，参见例子 3-26。

例子 3-26　input 函数的例子。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = input('随便输入数字：')
随便输入数字：123↙
A =
   123
>> whos
  Name        Size                    Bytes  Class
  A           1x1                         8  double array
Grand total is 1 element using 8 bytes
>> A = input('随便输入数字：', 's')
随便输入数字：123↙
A =
123
>> whos
  Name        Size                    Bytes  Class
  A           1x3                         6  char array
Grand total is 3 elements using 6 bytes
```

注意比较两次输入得到的变量 A 数据类型可以看出，在 input 函数中，第二个参数若指定为 s 的时候，则输入数据默认为字符串的格式。

3.5　元 胞 数 组

元胞数组是 MATLAB 的一种特殊数据类型，可以将元胞数组看作为一种无所不包的通用矩阵，或者叫作广义矩阵。组成元胞数组的元素可以是任何一种数据类型的常数或者常量，每一个元素也可以具有不同的尺寸和内存占用空间，每一个元素的内容也可以完全不同，所以元胞数组的元素叫作元胞(cell)。和一般的数值矩阵一样，元胞数组的内存空间也

是动态分配的。

与数值数组一致，元胞数组的维数不受限制，元胞数组可以是一维的、二维的，甚至也可以是多维的。访问元胞数组的元素可以使用单下标方式或者全下标方式。

3.5.1 元胞数组的创建

组成元胞数组的内容可以是任意类型的数据，所以创建元胞数组之前需要创建相应的数据。本小节结合具体的实例讲述创建元胞数组的方法和步骤。

例子 3-27(a) 创建元胞数组。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = {zeros(2,2,2) , 'Hello' ; 17.35,1:100}
A =
    [2x2x2 double]      'Hello'
    [         17.3500]    [1x100 double]
>> B = [{zeros(2,2,2)},{'Hello'};{17.35},{1:100}]
B =
    [2x2x2 double]      'Hello'
    [         17.3500]    [1x100 double]
>> C   = {1}
C =
    [1]
>> C(2,2) = {3}
C =
    [1]       []
     []      [3]
>> isequal(A,B)
ans =
     1
>> whos
  Name          Size                    Bytes  Class
  A             2x2                      1122   cell array
  B             2x2                      1122   cell array
  C             2x2                       144   cell array
  ans           1x1                         1   logical array
Grand total is 243 elements using 2389 bytes
```

创建元胞数组需要使用运算符花括号——“{}”，例如在创建数组 A 的时候，使用花括号将不同类型和尺寸的数据组合在一起构成了一个元胞数组，在这个数组中有标量、多维数组、向量和字符串。注意创建数组 B 时使用了不同的方法，该方法是将数组的每一个元素都使用花括号括起来，然后再用数组创建的符号方括号——“[]”将数组的元素括起来，

这时创建的数组 B 和前面创建的数组 A 完全一致，通过 isequal 函数的运行就可以看出。还有一种创建元胞数组的方法，如创建数组 C 时所用的方法，MATLAB 能够自动扩展数组的尺寸，没有被明确赋值的元素作为空元胞数组存在。

注意：

元胞数组占用的内存空间和元胞数组的内容相关，不同的元胞数组占用的内存空间不尽相同。另外，在显示元胞数组内容时，对于内容较多的元胞，显示的内容为元胞的数据类型和尺寸，例如显示数组 A 时的三维数组和长度为 100 的向量。另外仔细察看例子 3-27 创建元胞数组的不同方法。

一般的来说，构成元胞数组的数据类型可以是字符串、双精度数、稀疏矩阵、元胞数组、结构或其他 MATLAB 数据类型。每一个元胞数据也可以为标量、向量、矩阵、N 维数组。

例子 3-27(b)　创建元胞数组。

在 MATLAB 中还有一个函数——cell 可以用来创建元胞数组，本例子将使用这个函数创建元胞数组。在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = cell(1)
A =
    {[ ]}
>> B = cell(3,2)
B =
     [ ]     [ ]
     [ ]     [ ]
     [ ]     [ ]
>> C = cell(2,2,2)
C(:,:,1) =
     [ ]     [ ]
     [ ]     [ ]
C(:,:,2) =
     [ ]     [ ]
     [ ]     [ ]
>> whos
  Name      Size                    Bytes  Class
  A         1x1                         4  cell array
  B         3x2                        24  cell array
  C         2x2x2                      32  cell array
Grand total is 15 elements using 60 bytes
```

cell 函数的作用是用来创建空元胞数组，该函数可以创建一维、二维或者多维元胞数组，但是创建的数组都为空元胞。这里需要注意区别空数组和空元胞之间内存占用的区别，从例子 3-27(b)可以看出，元胞数组的每个空元胞占用四个字节的内存空间。

注意：

使用 cell 函数创建空元胞数组的主要目的是为数组预先分配连续的存储空间，节约内

存的占用，提高程序执行的效率。有关空元胞数组的元胞赋值将在 3.5.2 小节中讲述。

3.5.2　元胞数组的基本操作

所谓元胞数组的基本操作主要包括对元胞数组元胞和元胞数据的访问、修改，元胞数组的扩展、收缩或者重组。和操作一般的数值数组类似，操作数值数组的函数也可以应用在元胞数组上。本小节将结合具体的示例讲述元胞数组的基本操作。

例子 3-28(a)　元胞数组的访问。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = {zeros(2,2,2) , 'Hello' ; 17.35,1:10};
>> B = A(1,2)
B =
    'Hello'
>> class(B)
ans =
cell
>> whos
  Name      Size          Bytes  Class
  A         2x2             402  cell array
  B         1x1              70  cell array
  ans       1x4               8  char array
Grand total is 38 elements using 480 bytes
```

在例子 3-28(a)中，使用圆括号"( )"直接访问元胞数组的元胞，获取的数据也是一个元胞数组，尽管从表面上看来已经是字符串，但实际上是元胞数组，这一点请读者注意。

例子 3-28(b)　元胞元素的访问。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = {zeros(2,2,2) , 'Hello' ; 17.35,1:10};
>> C = A{1,2}
C =
Hello
>> class C
ans =
char
>> whos
  Name      Size          Bytes  Class
  A         2x2             402  cell array
  C         1x5              10  char array
  ans       1x4               8  char array
Grand total is 37 elements using 420 bytes
```

在例子 3-28(b)中，使用花括号“{}”可以直接获取元胞数组的元胞内容，和例子 3-28(a)比较，变量 B 的类型为元胞，但是变量 C 的类型为字符串，这就是访问元胞数组的两种操作符——“{}”和“()”之间的不同之处。

例子 3-28(c)　元胞元素的访问。

若需要访问元胞元素内部的成员，则需要将“{}”运算符和“()”结合起来使用。在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = {zeros(2,2,2) , 'Hello' ; 17.35,1:10};
>> D = A{1,2}(4)
D =
l
>> E = A{2,2}(5:end)
E =
     5     6     7     8     9    10
>> class(E)
ans =
double
>> F = A{4}([1 3 5])
F =
     1     3     5
>> whos
  Name      Size                    Bytes  Class
  A         2x2                       402  cell array
  D         1x1                         2  char array
  E         1x6                        48  double array
  F         1x3                        24  double array
  ans       1x6                        12  char array
Grand total is 44 elements using 488 bytes
```

将不同的括号——花括号、圆括号和方括号结合起来可以访问元胞元素的内部成员。特别是在例子 3-28(c)中，创建变量 F 的时候使用了三种括号访问了向量的元素。

元胞数组的扩充、收缩和重组的方法和数值数组大体相同，见下面的例子。

例子 3-29(a)　元胞数组的扩充。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = {zeros(2,2,2) , 'Hello' ; 17.35,1:10};
>> B = cell(2);
>> B(:,1) = {char('Hello', 'Welcome'); 10: - 1:5}
B =
     [2x7 char   ]     []
     [1x6 double]     []
>> C = [ A , B]
```

```
C =
    [2x2x2 double]    'Hello'         [2x7 char   ]    [ ]
    [     17.3500]    [1x10 double]   [1x6 double]     [ ]
>> D = [ A , B ; C]
D =
    [2x2x2 double]    'Hello'         [2x7 char   ]    [ ]
    [     17.3500]    [1x10 double]   [1x6 double]     [ ]
    [2x2x2 double]    'Hello'         [2x7 char   ]    [ ]
    [     17.3500]    [1x10 double]   [1x6 double]     [ ]
>> whos
  Name      Size                    Bytes  Class
  A         2x2                       402  cell array
  B         2x2                       204  cell array
  C         2x4                       606  cell array
  D         4x4                      1212  cell array
Grand total is 208 elements using 2424 bytes
```

重组或者收缩元胞数组的方法和数值数组相应的操作基本一致，见例子 3-29(b)。

例子 3-29(b)　元胞数组的收缩和重组。

接上例，在 MATLAB 命令行窗口中，键入下面的指令：

```
>> D(2,:) = []
D =
    [2x2x2 double]    'Hello'         [2x7 char   ]    [ ]
    [2x2x2 double]    'Hello'         [2x7 char   ]    [ ]
    [     17.3500]    [1x10 double]   [1x6 double]     [ ]
>> E = reshape(D,2,2,3)
E(:,:,1) =
    [2x2x2 double]    [17.3500]
    [2x2x2 double]    'Hello'
E(:,:,2) =
    'Hello'          [2x7 char]
    [1x10 double]    [2x7 char]
E(:,:,3) =
    [1x6 double]    [ ]
              [ ]   [ ]
>> whos
  Name          Size                    Bytes  Class
  A             2x2                       402  cell array
  B             2x2                       204  cell array
  C             2x4                       606  cell array
```

```
     D            3x4                          892  cell array
     E            2x2x3                        892  cell array
  Grand total is 270 elements using 2996 bytes
```

从例子 3-29(b)中可以看出，MATLAB 的元胞数组除了包含的数据类型有所区别外，其他的操作都和一般的数组无二。也就是说，操作元胞数组时，可以使用针对一般数组的操作方法。

3.5.3　元胞数组的操作函数

除了一般的元胞数组操作外，MATLAB 还提供了一部分函数用来进行元胞数组的操作，在表 3-11 中总结了这些函数。

表 **3-11**　元胞数组的操作函数

函　数	说　　明
cell	创建空的元胞数组
cellfun	为元胞数组的每个元胞执行指定的函数
celldisp	显示所有元胞的内容
cellplot	利用图形方式显示元胞数组
cell2mat	将元胞数组转变成为普通的矩阵
mat2cell	将数值矩阵转变成为元胞数组
num2cell	将数值数组转变成为元胞数组
deal	将输入参数赋值给输出
cell2struct	将元胞数组转变成为结构
struct2cell	将结构转变成为元胞数组
iscell	判断输入是否为元胞数组

下面结合一些具体的示例来讲解部分函数的使用方法。

例子 3-30　cellfun 函数示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = {rand(2,2,2), 'Hello', pi; 17, 1+i, magic(5)}
A =
    [2x2x2 double]     'Hello'                [    3.1416]
    [          17]     [1.0000+ 1.0000i]      [5x5 double]
>> B = cellfun('isreal', A)
B =
      1      1      1
      1      0      1
>> C = cellfun('length',A)
C =
      2      5      1
      1      1      5
>> D = cellfun('isclass', A,'double')
```

```
D =
     1     0     1
     1     1     1
>> whos
  Name        Size                    Bytes  Class
  A           2x3                       666  cell array
  B           2x3                         6  logical array
  C           2x3                        48  double array
  D           2x3                         6  logical array
Grand total is 65 elements using 726 bytes
```

从例子 3-30 中可以看出，cellfun 函数的主要功能是对元胞数组的元素(元胞)分别指定不同的函数，不过，能够在 cellfun 函数中使用的函数数量是有限的，见表 3-12。

表 **3-12** 在 **cellfun** 函数中可用的函数

函 数	说 明
isempty	若元胞元素为空，则返回逻辑真
islogical	若元胞元素为逻辑类型，则返回逻辑真
isreal	若元胞元素为实数，则返回逻辑真
length	元胞元素的长度
ndims	元胞元素的维数
prodofsize	元胞元素包含的元素个数

此外，cellfun 函数还有以下两种用法：

? cellfun('size', C, k)用来获取元胞数组元素第 k 维的尺寸。

? cellfun('isclass', C, classname)用来判断元胞数组的数据类型。

例子 3-31 显示元胞数组的内容。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = {rand(2,2,2),'Hello',pi;17,1+i,magic(5)}
A =
    [2x2x2 double]     'Hello'                [      3.1416]
    [           17]    [1.0000+ 1.0000i]      [5x5 double]
>> celldisp(A)
A{1,1} =
(:,:,1) =
          0.9355     0.4103
          0.9169     0.8936
(:,:,2) =
          0.0579     0.8132
          0.3529     0.0099
A{2,1} =
          17
```

```
A{1,2} =
Hello
A{2,2} =
          1.0000 + 1.0000i
A{1,3} =
          3.1416
A{2,3} =
          17    24     1     8    15
          23     5     7    14    16
           4     6    13    20    22
          10    12    19    21     3
          11    18    25     2     9
>> cellplot(A)
```

cellplot 函数的运行结果如图 3-3 所示。

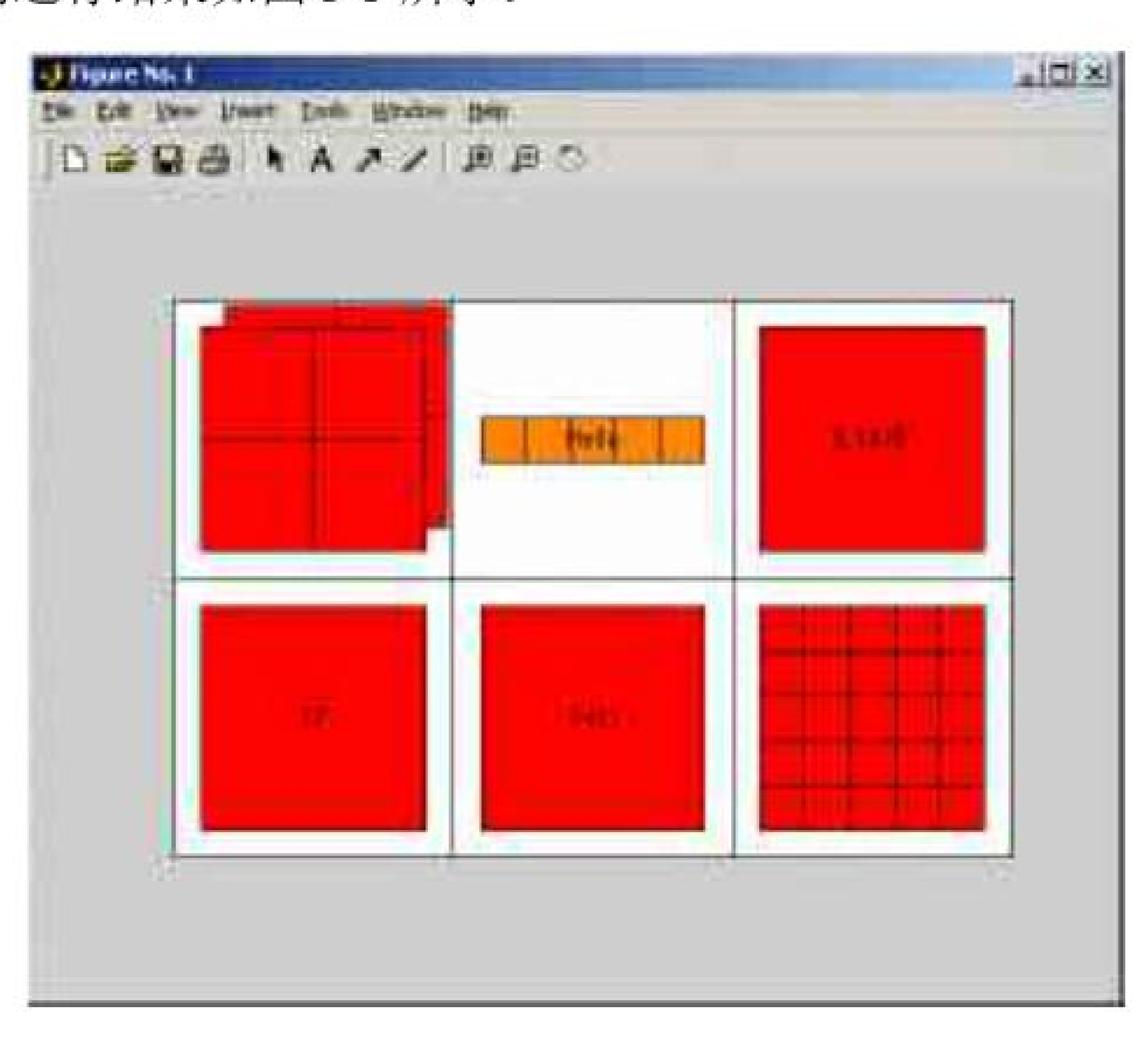


图 3-3　cellplot 函数的运行结果

cellplot 函数用图形的方式显示元胞数组的元素(元胞)，可以从图形中直观地观察元胞数组元素(元胞)的数据类型和简要的信息。而 celldisp 函数则是将元胞数组中所包含的内容显示在 MATLAB 命令行窗口中。

3.6　结　　构

和 C 语言类似，MATLAB 也具有结构类型的数据。结构(struct)是包含一组记录(records)的数据类型，而记录则存储在相应的字段(fields)中。和元胞数组类似，结构的字段可以是任意一种 MATLAB 数据类型的变量或者对象。结构类型的变量也可以是一维的、二维的或者

多维的数组。不过，在访问结构类型数据的元素时，需要使用下标配合字段的形式。

在 MATLAB 中，结构和元胞数组有诸多类似之处，在表 3-13 中进行了比较。

表 **3-13**　元胞数组和结构数组的异同

内容 \ 不同的数组对象	元胞数组对象	结构数组对象
基本元素	元胞(cell)	结构(struct)
基本索引	全下标方式、单下标方式	全下标方式、单下标方式
可包含的数据类型	任何数据类型	任何数据类型
数据的存储	元胞(cell)	字段(field)
访问元素的方法	花括号和索引	圆括号、索引和字段名

另外，在结构和元胞数组对象之间可以利用 MATLAB 的函数进行相互转换。

3.6.1　结构的创建

创建结构数组对象可以使用两种方法，一种是直接赋值的方法，另外一种方法是利用 struct 函数创建。这里结合具体的操作示例讲解创建结构的方法。

例子 3-32(a)　直接赋值法创建结构。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> Student.name = 'Way';
>> Student.age = 26;
>> Student.grade = uint16(1);
>> whos
  Name          Size                    Bytes  Class
  Student       1x1                       388  struct array
Grand total is 8 elements using 388 bytes
>> Student
Student =
        name: 'Way'
         age: 26
       grade: 1
```

在例子 3-32(a)中，创建了具有一个记录的 Student 结构数组，该数组具有一个元素(记录)，结构同时具有三个字段，分别为姓名(name)、年龄(age)和级别(grade)，这三个字段分别包含了字符串、双精度和无符号整数数据类型。

创建结构的时候，直接用结构的名称(如例子 3-32(a)中的 Student)，配合操作符“.”和相应字段的名称完成创建，创建是直接给字段赋具体的数值。

例子 3-32(b)　直接赋值法创建结构数组。

接例子 3-32(a)，在 MATLAB 命令行窗口中，键入下面的指令：

```
>> Student(3).name = 'Deni';
>> Student(3).grade = 2;
```

```
>> whos
  Name          Size                    Bytes  Class
  Student       1x3                       540  struct array
Grand total is 19 elements using 540 bytes
>> Student(2)
ans =
       name: [ ]
         age: [ ]
       grade: [ ]
>> Student(3).age
ans =
       [ ]
```

在例子 3-32(b)中，直接对结构数组的第三个记录的两个字段(name 和 grade)进行了赋值，则 MATLAB 将自动扩展结构数组的尺寸，对于没有赋值的字段，则直接创建空数组。

例子 3-33　利用函数 struct 创建结构。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> Student = struct('name', 'Way', 'age', 26, 'grade', uint16(1))
Student =
       name: 'Way'
        age: 26
      grade: 1
>> whos
  Name          Size                    Bytes  Class
  Student       1x1                       388  struct array
Grand total is 8 elements using 388 bytes
>> Student = struct('name', {'Deni', 'Sherry'}, 'age', {22,24}, 'grade', {2,3})
Student =
1x2 struct array with fields:
      name
      age
      grade
>> whos
  Name          Size                    Bytes  Class
  Student       1x2                       604  struct array
Grand total is 20 elements using 604 bytes
>> Student = struct('name', {}, 'age', {}, 'grade', {})
Student =
0x0 struct array with fields:
      name
      age
```

```
    grade
>> whos
  Name        Size                    Bytes  Class
  Student     0x0                       192  struct array
Grand total is 0 elements using 192 bytes
>> Student = repmat(struct('name', 'Way',' age', 26, 'grade', 1), 1, 3)
Student =
1x3 struct array with fields:
    name
    age
    grade
>> Student(3)
ans =
     name: 'Way'
      age: 26
    grade: 1
```

在例子 3-33 中讲述了使用 struct 函数创建结构的方法。struct 函数的基本语法为

```
struct_name = struct(field1,val1,field2,val2,……)
struct_name = struct(field1,{val1},filed2,{val2},……)
```

其中，这两种创建结构的方法在例子 3-33 中均有体现，而且利用 struct 函数还可以创建空结构数组。例子 3-33 最后使用了 repmat 函数，将结构制作了复本，从而创建了 1×3 结构数组。

3.6.2 结构的基本操作

对于结构的基本操作其实是对结构数组元素包含的记录的操作。主要有结构记录数据的访问、字段的增加和删除等。本小节结合具体的例子讲述有关结构操作的基本方法。

访问结构数组元素包含的记录的方法非常简单，直接使用结构数组的名称和字段的名称以及操作符“.”完成相应的操作。不过，在访问结构数组的元素时可以使用所谓的“动态”字段的形式，其基本语法结构是

```
struct_name.(expression)
```

其中，expression 是代表字段的表达式，可以是字段名称的字符串。利用动态字段形式访问结构数组元素，便于利用函数完成对结构字段数据的重复操作。

例子 3-34 结构字段数据的访问。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> Student = struct('name',{'Deni','Sherry'},'age',{22,24},'grade',{2,3},...
'score',{rand(3)*10,randn(3)*10});
>> %察看结构字段
>> Student
Student =
```

```
1x2 struct array with fields:
     name
     age
     grade
     score
>> %访问结构记录的数据
>> Student(2).score
ans =
   - 4.3256       2.8768       11.8916
   - 16.6558   - 11.4647     - 0.3763
     1.2533      11.9092       3.2729
>> %访问结构记录的一部分数据
>> Student(1).score(1,:)
ans =
     9.5013      4.8598      4.5647
>> %访问结构某一字段的所有数据
>> Student.name
ans =
Deni
ans =
Sherry
>> %使用动态字段形式访问数据
>> Student.('name')
ans =
Deni
ans =
Sherry
```

在例子 3-34 中使用了各种访问结构记录数据的方法，特别是在例子的最后，使用动态字段的形式访问了字段记录的数据。利用这种形式可以通过编写函数对结构记录的数据进行统一的运算操作。有关函数的编写请参阅本书的第四章的相关内容。

结构是管理数据的一种方式，在使用结构的时候，可以直接通过字段名称来访问数据，参见例子 3-35。

例子 3-35　对结构字段数据进行运算。

接例子 3-34，在 MATLAB 命令行窗口中键入下面的指令：

```
>> mean(Student(1).score)
ans =
      5.9604      7.1313      4.3213
>> mean(Student.score)
??? Error using ==> sum
Dimension argument must be a positive integer scalar.
```

```
Error in ==> I:\MATLAB6p5\toolbox\matlab\datafun\mean.m
On line 28   ==>   y = sum(x,dim)/size(x,dim);
>> mean([Student.score])
ans =
      5.9604      7.1313      4.3213     - 6.5761      1.1071      4.9294
```

mean 函数为 MATLAB 内建的数学函数之一，用来求解列向量的平均值。

在例子 3-35 中，首先对结构数组第一个元素的字段 score 代表的数据求平均值，其实该操作和使用 MATLAB 普通的变量没有什么区别。然后后面的操作就专属于结构了。注意，最后的操作相当于对结构数组的某一个字段所有的数据进行同一种操作，不过，这个时候需要使用“[]”符号将字段包含起来，否则会出现错误提示。

如前文所述，MATLAB 结构数组的字段可以包含任何一种数据，自然也可以包含结构。当结构的字段记录了结构时，则称其为内嵌(nest)的结构。创建内嵌结构可以使用直接赋值的方法，同样也可以使用 struct 函数完成。

例子 3-36　内嵌的结构。

```
>> Student = struct('name',{'Deni','Sherry'},'age',{22,24},'grade',{2,3},...
'score',{rand(3)*10,randn(3)*10});
>> Class.number = 1;
>> Class.Student = Student;
>> whos
  Name          Size                    Bytes  Class
  Class         1x1                      1188  struct array
  Student       1x2                       932  struct array
Grand total is 83 elements using 2120 bytes
>> Class
Class =
      number: 1
      Student: [1x2 struct]
>> %使用 struct 函数创建内嵌的结构
>> Class = struct('number',1,'Student',struct('name',{'Way','Deni'}))
Class =
      number: 1
      Student: [1x2 struct]
```

在例子 3-36 中使用了两种不同的方法创建内嵌结构。访问内嵌结构数据的方法和访问普通结构字段的方法类似，只要将内嵌的结构看作结构的元素之一就可以了，具体的方法这里就不再赘述了，请读者自己实践掌握。

3.6.3　结构操作函数

和其他的各种数据类似，MATLAB 也提供了部分函数用于针对结构的操作，在表 3-14 中将这些函数进行了总结。

表 **3-14**　结构操作函数

函　数	说　　明
struct	创建结构或将其他数据类型转变成结构
fieldnames	获取结构的字段名称
getfield	获取结构字段的数据
setfield	设置结构字段的数据
rmfield	删除结构的指定字段
isfield	判断给定的字符串是否为结构的字段名称
isstruct	判断给定的数据对象是否为结构类型
oderfields	将结构字段排序

除了上述这些函数之外，在元胞数组一节讲述的部分函数也可以应用在结构数据对象中，本小节将结合部分例子讲解部分函数的用法。

例子 3-37　结构操作函数的使用示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> S.name = 'Deni';S.ID = 0;
>> S(2,2).name = 'Way';S(2,2).ID = 1;
>> S2 = setfield(S,{2,1},'name','Sherry');
>> S.name
ans =
Deni
ans =
        []
ans =
        []
ans =
Way
>> S2.name
ans =
Deni
ans =
Sherry
ans =
        []
ans =
Way
>> fieldnames(S)
ans =
      'name'
      'ID'
```

```
>> S3 = orderfields(S)
S3 =
2x2 struct array with fields:
    ID
    name
```

在例子 3-37 中，主要讲述了 setfield、fieldnames 和 orderfields 函数的使用。setfield 函数是为结构字段进行赋值的函数，对应的可以使用 getfield 函数获取结构字段的数值。利用 setfiled 函数和 struct 函数可以有效地创建结构数组。fieldnames 函数用来获取结构中的字段名称，由字段的名称组成元胞数组，其中元胞就是字段名称字符串。orderfileds 函数是用来将字段进行排序的，该函数能够将结构的字段按照字符序号排列。例如在例子 3-37 中，orderfileds 函数就将字段 ID 和 name 进行了排序，该函数不会修改结构中包含的内容。

例子 3-38 结构操作函数的使用示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> S.name = 'Deni';S.ID = 0;
>> S(2).name = 'Way';S(2).ID = 1;
>> C = struct2cell(S)
C(:,:,1) =
    'Deni'
    [    0]
C(:,:,2) =
    'Way'
    [   1]
>> C = squeeze(C)
C =
    'Deni'      'Way'
    [    0]     [   1]
>> fields = {'Name','ID'};
>> S2 = cell2struct(C',fields,2)
S2 =
2x1 struct array with fields:
    Name
    ID
>> whos
  Name        Size                    Bytes  Class
  C           2x2                       270  cell array
  S           1x2                       398  struct array
  S2          2x1                       398  struct array
  fields      1x2                       132  cell array
Grand total is 47 elements using 1198 bytes
```

例子 3-38 主要演示了 cell2struct 函数和 struct2cell 函数的使用方法。从表现形式、用途以及保存的数据类型等方面看，结构和元胞非常的类似，因此 MATLAB 提供了将这两种数据类型进行相互转换的函数。在将元胞数组转换为结构的时候需要指定结构的字段名称。函数 squeeze 是用来删除多维数组中维数为 1 的那一维的函数。

例子 3-39　结构操作函数的使用示例。

```
>> clear all
>> X = 3;
>> [y1,y2,y3] = deal(X)
y1 =
     3
y2 =
     3
y3 =
     3
>> X = {rand(3),'2',1};
>> [y1,y2,y3] = deal(X{:})
y1 =
    0.8462    0.6721    0.6813
    0.5252    0.8381    0.3795
    0.2026    0.0196    0.8318
y2 =
    2
y3 =
     1
>> X .num = rand(3);X.str = '2'; X.ID = 1;
>> X(2).num = rand(3);X(2).str = '3'; X(2).ID = 2;
>> [y1,y2] = deal(X(:).num)
y1 =
    0.5028    0.3046    0.6822
    0.7095    0.1897    0.3028
    0.4289    0.1934    0.5417
y2 =
    0.1509    0.8600    0.4966
    0.6979    0.8537    0.8998
    0.3784    0.5936    0.8216
```

例子 3-39 主要演示了函数 deal 的使用方法。deal 函数的主要作用是进行数据的复制。该函数不仅能够操作一般的数值类型数据，还可以处理结构或者元胞。在例子 3-39 中，deal 函数处理了标量、元胞和结构。在 deal 函数处理标量的时候，函数将标量的数值依次赋值给函数的输出，所以第一次使用 deal 函数的时候 y1、y2 和 y3 都被赋值为数值 3。不过在处

理元胞或者结构的时候，该函数执行的效果则不同。它能够将元胞数组中的元胞或者结构中的某个字段的数据依次赋值给相应的输出，所以利用该函数可以非常方便地将结构或者元胞中的实际数据解析(分离)出来，例如在例子 3-39 中所完成的操作。

关于本小节提及的函数的详细解释，请读者自行查阅 MATLAB 的帮助文档或者在线帮助。

3.7 本 章 小 结

本章集中介绍了 MATLAB 的各种数据类型，其中包括了基本的数值类型、逻辑类型数据、元胞数组、结构和字符串等，同时介绍了部分处理这些不同类型数据变量的函数。本章以及第二章的内容都是后面章节中进一步学习使用 MATLAB 的基础。MATLAB 作为一种开发环境，一种计算软件，其数据管理和维护的能力非常强，所以，在进行后面章节的学习之前，读者一定要充分地掌握这些数据类型的基本用法。掌握这些基本的数据类型，不仅能够便于读者快速入门学习 M 语言编程，灵活合理地使用不同的数据类型和相关的函数，而且还能够有效地提高编写和执行 MATLAB 应用程序的效率。因此请读者一定要熟练掌握本章的内容，这样，在后面的学习过程中就会有一种轻车熟路的感觉了。

第四章　MATLAB 编程基础

在前面的章节中已经介绍了在 MATLAB 中利用 M 语言进行编程的基本要素——各种数据类型和相应的数据对象的声明方法。本章将详细讲解利用 M 语言进行编程的方法。

本章讲述的主要内容如下：

- 流程控制；
- 脚本文件；
- 函数文件；
- 子函数；
- 私有函数；
- □文件的调试。

4.1　概　　述

MATLAB 提供了完整的编写应用程序的能力，这种能力通过一种被称为 M 语言的高级语言来实现。这种编程语言是一种解释性语言，利用该语言编写的代码仅能被 MATLAB 接受，被 MATLAB 解释、执行。其实，一个 M 语言文件就是由若干 MATLAB 的命令组合在一起构成的，这些命令都是在前面章节中介绍的合法的 MATLAB 命令。和 C 语言类似，M 语言文件都是标准的纯文本格式的文件，其文件的扩展名为.m。

使用 M 文件最直接的好处就是可以将一组 MATLAB 命令组合起来，通过一个简单的指令就可以执行这些命令。这些命令可以完成某些 MATLAB 的操作，也可以实现某个具体的算法。其实，MATLAB 产品族中包含的工具箱就是由世界上在相应专业领域内的顶尖高手，利用 M 语言开发的算法函数文件集合。读者也可以结合自己工作的需要，为自己的 MATLAB 开发具体的算法和工具箱。

MATLAB 的函数主要有两类，一类被称为内建(Build-in)函数，这类函数是由 MATLAB 的内核提供的，能够完成基本的运算，例如三角函数、矩阵运算的函数等。另外一类函数就是利用高级语言开发的函数文件，这里的函数文件既包括用C语言开发的MEX函数文件，又包含了 M 函数文件。有关 MEX 函数文件的内容已经超出了本书的内容，将在《MATLAB 外部编程接口》一书中详细讲述。

如前所述，MATLAB 的 M 语言文件是纯文本格式的文件，利用任何一种纯文本编辑器都可以编写相应的文件，例如 Windows 平台下的记事本、UltraEdit 等软件，或者 Unix 平台下的 Emacs 软件等。同样，为了方便编辑 M 文件，MATLAB 也提供了一个编辑器，叫作 meditor，它也是系统默认的 M 文件编辑器。

运行 meditor 的方法非常简单，在 MATLAB 命令行窗口中键入下面的指令就可以打开 meditor：

```
>> edit
```

这时 MATLAB 将启动 meditor，然后创建一个未命名的空白文件，如图 4-1 所示。

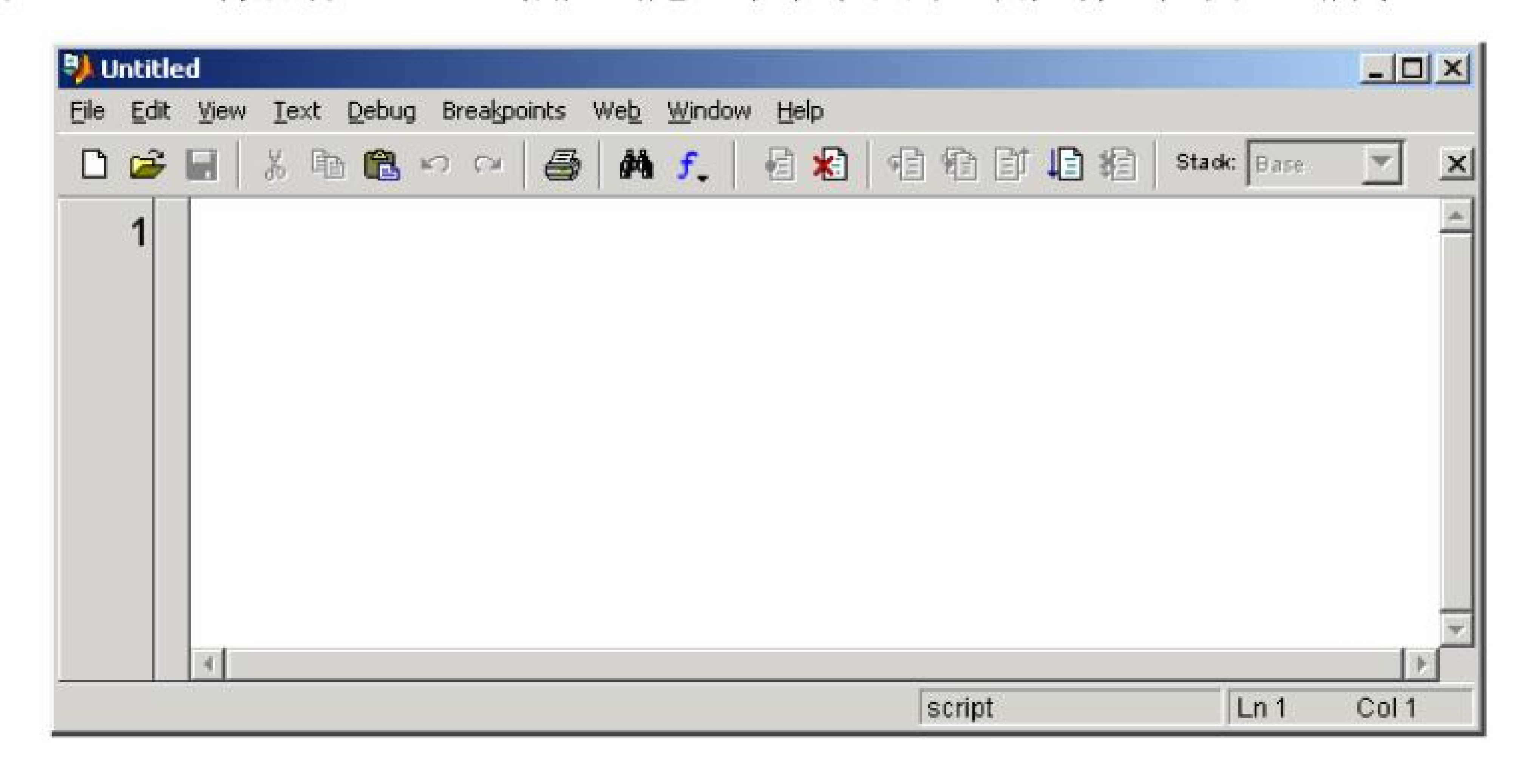


图 4-1 meditor 的运行界面

这时用户就可以直接在编辑器中键入 MATLAB 指令，开发 M 语言文件了。

此外，运行 meditor 还可以通过“File”菜单中“New”子菜单下的“M-File”命令来实现，或者直接单击 MATLAB 用户界面工具栏上的新建按钮完成同样的工作。

M 语言文件可以分为两类，其中一类是脚本文件，另外一类叫作函数文件。本章将分别介绍这两类文件的编写方法。

4.2 流 程 控 制

程序流程控制包含控制程序流程的基本结构和语法，例如应用程序的选择和循环结构，这也是结构化编程的基本结构，使用结构化的应用程序设计方法可以使设计的程序结构清晰、可读性强，能够提高应用程序的设计效率，增强程序的可维护性。结构化应用程序的设计思想是现代程序设计的基础。结构化的程序主要有三种基本的程序结构：顺序结构、选择结构、循环结构。

所谓顺序结构就是指所有组成程序源代码的语句按照由上至下的次序依次执行，直到程序的最后一个语句，也就是程序语句的简单罗列。而选择结构是依照不同的判断条件进行判断，然后根据判断的结果选择某一种方法来解决某一个问题。循环结构就是在程序中某一条语句或者多条语句重复多次的运行结构。

已经得到证明，上述的三种程序结构足以处理各种各样的复杂问题，将上述的三种结构组合在一起就可以构成更复杂的程序。而一般的 M 语言文件都是由上述三种结构的 MATLAB 指令构成的。

4.2.1　选择结构

如前所述，当人们判断某一条件是否满足，根据判断的结果来选择不同的解决问题的方法时，就需要使用选择结构。和 C 语言类似，MATLAB 的条件判断可以使用 if 语句或者 switch 语句。

4.2.1.1　if 语句

if 语句的基本语法结构有三种，分别如下：

(1) if (关系运算表达式)

　　MATLAB 语句

end

这种形式的选择结构表示，当关系运算表达式计算的结果为逻辑真的时候，执行 MATLAB 语句，这里的 MATLAB 语句可以是一个 MATLAB 表达式，也可以是多个 MATLAB 表达式。在 MATLAB 语句的结尾处，必须有关键字 end。

(2) if(关系运算表达式)

　　MATLAB 语句 A

else

　　MATLAB 语句 B

end

这种选择结构表示，当关系运算表达式的计算结果为逻辑真的时候，执行 MATLAB 语句 A，否则执行 MATLAB 语句 B，在语句 B 的结尾必须具有关键字 end。

(3) if (关系运算表达式 a)

　　MATLAB 语句 A

　elseif (关系运算表达式 b)

　　MATLAB 语句 B

　else (关系运算表达式 c)

　⋮

end

这种选择结构可以判断多条关系运算表达式的计算结果，然后按照执行的逻辑关系执行相应的语句。读者可以根据类似的 C 语言知识或者前面两种选择结构的介绍判断这种结构的执行方式。

例子 4-1　if 语句的使用——if_examp.m。

读者通过本例子将同时了解 meditor 的基本使用方法。打开 meditor，然后键入下面的指令：

```
clear all

I=1;
J=2;

```

```
006	if I == J
007		A(I,J) = 2;
008	elseif abs(I-J) == 1
009		A(I,J) = - 1;
010	else
011	A(I,J) = 0;
012	end
```

注意：

在键入程序时，不要将行号(001～012)也敲进去，在这里设置行号的主要目的是为了便于讲解和分析程序。

所有的指令键入完毕后，将文件保存，读者可以将其保存为任何名字，不过文件名必须由英文字符和数字组成，将文件的扩展名设置为.m，并且将文件保存在 MATLAB 的搜索路径下，例如 MATLAB 当前的工作路径。

然后在 MATLAB 的命令行中，键入刚才保存的文件名，不过，这时不要将扩展名也一同键入，MATLAB 就会依次执行这些指令。

运行例子 4-1 的方法和效果如下：

```
>> if_examp
A =
     0    - 1
```

例子 4-1 代码的核心是 006～012 行的部分，这部分展示了 if-elseif-else-end 语句组合的使用方法。请读者仔细察看，并且通过修改程序 003 和 004 行中对 I 和 J 的赋值来察看整个语句的执行情况。

和 C 语言类似，if-elseif-else 的语句结构也可以嵌套地使用，也就是可以存在这样的语句结构：

```
if(关系表达式 a)
	if(关系表达式 b) MATLAB 语句 A
	else   MATLAB 语句 B
	end
else
	if(关系表达式 c) MATLAB 语句 C
	else   MATLAB 语句 D
	end
end
```

注意：

在使用嵌套的选择结构时，需要小心 if 语句和 end 关键字的配对。

例子 4-2　嵌套使用的 if 结构——if_examp2.m。

```
clear all

if 1
```

```
004         disp('Is 1')
005     else
006         disp('Not 1')
007     end
008
009     I = 1;
010     if I
011         if I < 2
012             disp('I is bigger than 0 but less than 2')
013         end
014     else
015         if I > - 2
016             dis('I is less than 0 but bigger than - 2')
017         end
018     end
```

该程序的运行方法和效果如下：

```
>> if_examp2
Is 1
I is bigger than 0 but less than 2
```

在例子 4-2 中，主要说明了嵌套的 if 结构和在关系表达式中使用常量的方法。在代码的 003 行，if 语句的关系表达式为常数 1，这个时候 if 语句将始终认为非零值为逻辑真，所以，程序执行了 004 行的代码。同样，在程序的 009 行，if 语句的关系表达式为变量 I，若 I 的数值为非零值，则 if 语句判断其为逻辑真，所以，代码的 016 行只有在 I 为 0 时，才可能被执行。

4.2.1.2　switch 语句

另外一种构成选择结构的关键字就是 switch。在处理实际问题的时候，往往要处理多个分支，这时如果使用 if-else 语句处理多分支结构往往使程序变得十分冗长，从而降低了程序的可读性。switch 语句就可以用于处理这种多分支的选择，它的基本语法结构如下：

```
switch(表达式)
    case 常量表达式 a：MATLAB 语句 A
    case 常量表达式 b：MATLAB 语句 B
    ⋮
    case 常量表达式 m：MATLAB 语句 M
    otherwise         ：MATLAB 语句 N
end
```

在 switch 语句之后的表达式可以是一个数值类型表达式或者是一个数值类型的变量，当这个表达式的值同 case 后面的某一个常量表达式相等时，则执行该 case 后面的常量表达式后面的语句。

注意：

MATLAB 的 switch 和 C 语言的 switch 语句结构不同。在 C 语言中，每一个 case 后面的语句中必须包含类似 break 语句的流程控制语句，否则程序会依次执行符合条件的 case 语句后面的每一个 case 分支。但是在 MATLAB 中就不必如此，程序仅仅执行符合条件的 case 分支。

例子 4-3　switch 结构使用示例——switch_examp.m。

```
clear all

algorithm = input('Enter an algorithm in quotes (ode23, ode15s, etc:) ');

switch algorithm
case 'ode23'
       str = '2nd/3rd order';
case {'ode15s', 'ode23s'}
    str = 'stiff system';
otherwise
str = 'other algorithm';
end
disp(str);
```

该文件的运行方法和效果如下：

```
>> switch_examp
Enter an algorithm in quotes (ode23, ode15s, etc:) 'ode23'
2nd/3rd order
>> switch_examp
Enter an algorithm in quotes (ode23, ode15s, etc:) 'ode4'
other algorithm
```

例子 4-3 中需要用户在执行程序的过程中输入一个字符串，switch 语句根据用户的输入判断执行相应的 case 分支。若没有符合条件的 case 分支，则 switch 执行 otherwise 后面的语句。若 switch 结构中没有定义 otherwise 及其相应的代码，则程序不会进行任何操作，而是直接退出 switch 结构。

提示：

在处理以字符串变量或者常量参与的关系判断操作时，使用 switch 结构要比 if-else 结构效率高一些。

由于 MATLAB 的 switch 结构没有 C 语言的 fall-through 特性，所以，如果需要针对多个条件而使用同一个 case 分支的时候，需要使用元胞数组与之配合，参见例子 4-4。

例子 4-4　switch 结构使用示例——switch_examp2.m。

```
clear all

var = input('Input a Numer:');
```

```
004    switch var
005        case 1
006            disp('1')
007        case {2,3,4}
008            disp('2 or 3 or 4')
009        case 5
010            disp('5')
011        otherwise
012            disp('something else')
013    end
```

例子 4-4 运行的方法和效果如下：

```
>> switch_examp2
Input a Numer:1
1
>> switch_examp2
Input a Numer:3
2 or 3 or 4
>> switch_examp2
Input a Numer:7
something else
```

例子 4-4 代码的核心部分为 007 行，这里使用元胞数组增加判断条件的个数，当输入的数字为 2、3 或者 4 时，switch 结构将使用同一个 case 分支进行判断、计算。

注意：

从代码的完整性和可靠性角度出发，在使用 switch 语句时，一定要包含 otherwise 分支，这是一种良好的编程习惯。

4.2.2 循环结构

在解决很多问题的时候需要使用循环结构，例如求解数列的和或者进行某种迭代法求解数值方程时，都需要循环结构配合完成计算。

在 MATLAB 中，包含两种循环结构，一种是循环次数不确定的 while 循环，而另一种是循环次数确定的 for 循环。

1．while 循环结构

while 语句可以用来实现“当”型的循环结构，它的一般形式如下：

```
while(表达式)
MATLAB 语句
end
```

当表达式为真时，循环将执行由语句构成的循环体，其特点是先判断循环条件，如果循环条件成立，即表达式运算结果为“真”，再执行循环体。循环体执行的语句可以是一句

也可以是多句，在 MATLAB 语句之后必须使用关键字 end 作为整个循环结构的结束。另外，在循环过程中一定要能够改变关系表达式或者布尔类型变量的值，或者使用其他方法来跳出循环，否则会陷入死循环(无法正常退出的循环叫作死循环)。

例子 4-5 使用 while 语句求解 $\sum_{n=1}^{1000} n$。

```
001    i = 1;
002    sum = 0;
003    while ( i <= 1000 )
004        sum = sum+i;
005        i = i+1;
006    end
007    str = ['计算结果为：',num2str(sum)];
008    disp(str)
```

例子 4-5 的运行结果为

```
>> while_example
计算结果为：500500
```

例子 4-5 的 002~006 行使用了 while 循环结构，在循环结构中进行了累加的操作。需要注意的是，在 MATLAB 中没有类似 C 语言的++或者+=等运算操作符，因此在进行诸如累加或者递减的运算时，不得不给出完整的表达式。另外，例子 4-5 求数列和的算法的运算效率很低，在 MATLAB 中不要使用这样的结构完成类似的运算，而需要采用向量化的计算。

注意：

while 循环结构的关系表达式可以是某个数据变量或者常量，这时，将按照非零值为逻辑真进行相应的操作。另外，在进行上述操作时，若数据变量为空矩阵，则 while 语句将空矩阵作为逻辑假处理，也就是说，在 while A　MATLAB 语句 S1　end 结构中，若 A 为空矩阵，则 MATLAB 语句 S1 永远不会被执行。

2．for 循环结构

使用 for 语句构成循环是最灵活、简便的方法，不过，使用 for 语句循环需要预先知道循环体执行的次数，所以这种循环一般叫作确定循环。在 MATLAB 中 for 循环的基本结构如下：

```
for index = start:increment:end
   MATLAB 语句
end
```

其中，index 的取值取决于 start 和 end 的值，一般地，这里通常使用等差的数列向量，参见例子 4-6。

例子 **4-6** 使用 for 语句求解 $\sum_{n=1}^{1000} n$。

```
sum = 0;
```

```
002    for i = 1:1000
003         sum = sum+i;
004    end
005    str = ['计算结果为：',num2str(sum)];
006    disp(str)
```

例子 4-6 运行的结果为

```
>> for_example
计算结果为：500500
```

在例子 4-6 中，002 行的代码使用了确定次数的 for 循环结构，循环次数使用行向量进行控制，而且索引值 i 按照默认的数值 1 进行递增。

在 for 循环语句中，不仅可以使用行向量进行循环迭代的处理，也可以使用矩阵作为循环次数的控制变量，这时循环的索引值将直接使用矩阵的每一列，循环的次数为矩阵的列数，参见例子 4-7。

例子 **4-7**　for 循环示例。

```
001    A = rand(3,4);
002
003    for i = A
004         sum = mean(i)
005    end
```

例子 4-7 运行的结果为

```
>> for_matrices
sum =
      0.2728
sum =
      0.6649
sum =
      0.4275
sum =
      0.5220
```

例子 4-7 尽管只有短短的几行，但是在 003 行使用了一个矩阵作为循环的索引值，于是，循环结果就分别计算矩阵的每一列元素的均值。

和其他高级语言类似，MATLAB 的循环结构也可以进行嵌套使用，使用嵌套的循环需要注意 for 关键字和 end 关键字之间的配对使用，请读者根据高级语言的一般特性来推断其运行的方式，这里就不再赘述了。

4.2.3　break 语句和 continue 语句

在循环结构中还有两条语句会影响程序的流程，这就是 break 语句和 continue 语句，这两条语句的基本功能如下：

? 当 break 语句使用在循环体中的时候，其作用是能够在执行循环体的时候强迫终止循环，即控制程序的流程，使其提前退出循环，它的使用方法是

break;

? continue 语句出现在循环体中的时候，其作用是能够中断本次的循环体运行，将程序的流程跳转到判断循环条件的语句处，继续下一次的循环，它的使用方法是

continue;

下面结合具体的例子说明这两种不同语句的使用方法。

例子 4-8 break 语句示例——break_example.m。

```
i = 0;
j = 0;
k = 0;
for i = 1:2
    for j = 1:2
        for k = 1:2
            if(k == 2)
                disp('退出循环');
                break;
            end
            str = sprintf('I = %d , J = %d , K = %d',i,j,k);
            disp(str);
        end
    end
end
disp('程序运行结束');
```

例子 4-8 的运行结果如下：

```
>> break_example
I = 1 , J = 1 , K = 1
退出循环
I = 1 , J = 2 , K = 1
退出循环
I = 2 , J = 1 , K = 1
退出循环
I = 2 , J = 2 , K = 1
退出循环
程序运行结束
```

break 语句的作用是退出当前的循环结构运行，所以在例子 4-8 中，位于最内层循环的 break 语句执行的结果是退出了最内层的循环 k，位于外层的循环 i 和 j 还是都运行完毕了。

例子 4-9　continue 语句示例。

```
i = 0;
for i =1:6
    if(i>3)
        continue
    else
        str = sprintf('I = %d',i);
        disp(str);
    end
end
str = sprintf('循环结束  I = %d',i);
disp(str);
```

例子 4-9 的运行结果如下：

```
>> continue_example
I = 1
I = 2
I = 3
循环结束  I = 6
```

continue 语句的作用在例子 4-9 中得到了充分说明，该语句终止当前的循环，然后继续下一次循环运算，直到所有的循环迭代运算结束为止。

注意：

在程序中使用 break 和 continue 语句来改变程序的流程不是非常好的编程习惯，所以在编写程序的时候，尽量不要使用上述两条语句。

4.2.4　提高运算性能

M 语言和其他的高级语言不同，由于采用了解释型语言，所以 M 语言的执行效率肯定低于编译型语言(例如 C 语言)。然而，随着 MATLAB 版本的不断升级，再加之合理利用 MATLAB 向量运算等特点可以较大幅度地提高 M 语言代码的执行效率。在本小节结合一些具体的例子来讲述 M 语言编程以及 MATLAB 软件本身在提高程序执行效率方面的一些特性。

提示：

尽管 M 语言的执行效率无法与 C 语言或者 Fortran 语言等相媲美，但是，由于 M 语言依托于 MATLAB 强大的数值计算和分析能力，有众多的工具箱函数可以直接使用，因此，从程序的开发效率上讲，它是工程领域内最方便、快捷的编程语言。因此，如果读者主要完成的工作是进行算法开发与验证，那么 M 语言就成为一种最佳的选择。

1．向量化运算

首先，希望读者牢记这样一点，MATLAB 最初的目的是提供便利的矩阵数据操作能力。所以在大多数的应用程序中，不要使用循环结构操作矩阵的元素，应直接使用矩阵元素的

索引或者矩阵运算的函数，这样做不仅能够提高代码的执行效率，而且还能够提高程序的可读性，这就是所谓的向量化的运算，也就是说，尽量将使用 while 循环或者 for 循环的语句结构转换成等价的向量或者矩阵运算，以提高程序的运算速度，参见例子 4-10。

例子 4-10 向量化运算——array_vs_loops.m。

```
001 Mass = rand(5,10000);
002 Length = rand(5,10000);
003 Width = rand(5,10000);
004 Height = rand(5,10000);
005
006 [rows, cols] = size(Mass);
007
008 disp([char(10), '使用数组运算：'])
009 tic
010 Density = Mass./(Length.*Width.*Height);
011 toc
012
013 disp([char(10), '使用循环结构：'])
014 tic;
015 for I = 1:rows
016     for J = 1:cols
017         Density(I) = Mass(I,J)/(Length(I,J)*Width(I,J)*Height(I,J));
018     end
019 end
020 toc
```

例子 4-10 比较了循环结构和数组运算的执行效率，程序中分别在 010 行使用数组运算和在 015～019 行使用循环结构完成了同样的工作。程序的运行结果如下：

```
>> array_vs_loops
使用数组运算：
elapsed_time =
          0
使用循环结构：
elapsed_time =
          0.0100
```

通过程序运行的结果可以看出数组运算和循环迭代结构在计算效率方面的差距，特别是在循环迭代层次较多的时候，数组运算的速度优势就更加明显。所以，在大多数情况下，需要尽量将循环迭代工作改写成数组或者矩阵的运算，以提高程序的执行效率。

2．预分配存储空间

另外一种能够提高运算效率的方法就是进行内存变量存储空间的预分配，首先察看例子 4-11。

例子 4-11　内存预分配的例子—— pre_allocate.m。

```
001    disp([char(10), '使用内存预分配：'])
002    pre_allo = zeros(10000,1);
003    tic;
004    for I = 1:10000
005        pre_allo(I) = rand(1);
006    end
007    toc
008
009    disp([char(10), '不使用内存预分配：'])
010    tic;
011    for J = 1:10000
012        not_pre_allo(J) = rand(1);
013    end
014    toc
```

例子 4-11 的执行结果如下：

```
>> pre_allocate
使用内存预分配：
    elapsed_time =
              0.0900
不使用内存预分配:
    elapsed_time =
              0.3410
```

上面两种不同的运算惟一的区别就是程序 002 行，执行这行语句之后，MATLAB 自动分配了 10 000 个连续的内存空间用于存储数据，MATLAB 将一次创建足够的存储空间，然后依次赋值。而后者 not_pre_alloc 变量没有进行相应的操作，所以带来了两次运算结果的不同。

在不使用内存预分配的运算中，MATLAB 是如何进行操作的呢？

当 I=1 时，MATLAB 将使用一小块长度为一个单元大小的内存保存一位随机数。

当 I=2 时，MATLAB 寻找一块两单元大小的内存区，一个单元放第一个随机数，第二个放另外一个随机数。

⋮

当 I = 10000 时，MATLAB 寻找一块容纳 10000 单元的内存区存放以前的 9999 个随机数，同时把最新的一个随机数加入进去。代码运行的结果造成了存储空间的浪费，降低了程序的执行速度。

所以，在编写 M 语言程序的时候需要尽量使用内存的预分配，而少使用或者不使用数组内存空间的自动扩充方式。MATLAB 针对不同的数据类型有不同的内存预分配函数，见表 4-1。

表 **4-1** 内存预分配函数

数据类型	函 数	例 子
数值数组	zeros	Y = zeros(1:10000)
元胞数组	cell	Y = cell(2,3); Y{1,3} = zeros(1:10000); Y{2,3} = 'string'
结构数组	struct、repmat	Y = repmat(struct(field,value),2,3)

表 4-1 中说明了不同数据类型所要使用的内存预分配函数，其中结构类型的数组需要两个函数配合使用，利用 struct 函数构造结构，而使用 repmat 函数创建数组。

对于非双精度类型的数据，例如整数类型或者单精度类型，进行内存的预分配时，需要使用相应的构造函数或者类型转换函数，例如：

```
Y = int16(zeros(1:10000));
```

在上面的表达式中创建了连续的 10 000 个 16 位整数的存储空间。

当预先分配的内存空间无法容纳数据时，则可以通过 repmat 函数来扩充数组的存储空间。

上述函数的具体使用方法请读者参阅 MATLAB 的帮助文档或者在线帮助。

3．MATLAB 加速器

尽管利用了上述的特性可以明显提高 M 语言的运行效率，但是在同等条件下，M 语言的效率还是无法和第四代高级编程语言的执行效率相媲美，于是，在 MATLAB 6.5 版本中，新增加了 MATLAB 性能加速器这样一个新特性。MATLAB 性能加速器能够明显改善循环结构、内建函数调用等方面的运行效率，特别是在处理循环迭代次数确定的 for 循环结构中，其运算速度较早期的 MATLAB 版本有很大的提高，在表 4-2 中总结了 MATLAB 加速器能够发挥作用的 M 语言元素。

表 **4-2** 加速器能够影响的 **M** 语言元素

M 语言元素	作 用 范 围 说 明
数据类型	MATLAB 性能加速器能够针对布尔类型(逻辑数组)、字符串类型、8 位、16 位和 32 位整数类型(包含有符号和无符号整数类型)、双精度类型的数据运算进行性能加速
数组构造	MATLAB 性能加速器可以针对数组的构造(shape)进行加速，但是对于维数超过三的多维数组无能为力
for 循环结构	对于具备下列特征的 for 循环结构能够进行性能加速： 循环迭代的索引是标量； 循环体是加速器支持的数据类型运算和(或)数组构造重组操作； 循环体内调用的函数都是 MATLAB 的内建函数
选择结构	对于关系表达式使用标量运算的选择结构，MATLAB 性能加速器能够提高其执行的速度

在 MATLAB 中也有一部分 M 语言元素不能被 MATLAB 性能加速器加速，这些元素在表 4-3 中进行了总结。

表 **4-3**　性能加速器不起作用的 **M** 语言元素

M 语言元素	说　明
数据类型	除了能够加速的数据类型以外，其他的数据类型例如单精度类型、元胞数组、结构数组、Java 类对象等数据参与的操作无法加速
函数调用	若在 MATLAB 代码中调用了其他 M 或 MEX 函数，则性能加速器无法发挥作用
多操作代码行	若在一行代码中进行了多条语句操作，例如 x = a.name; for k=1:10000, sin(A(k)), end;这样的语句无法进行加速
数据类型改变	若在程序中对已经存在的变量修改了数据类型，则性能加速器不能发挥作用，例如 X = 23; -accelerated code- X = 'A';%这行代码不能加速 -more accelerated code-
复数常量	若在运算中直接将 i 和 j 作为复数常量参与运算，例如 Y=2+3*i，则性能加速器不能发挥作用，正确的做法应该是写作 Y= 2+3i

为了能够说明 MATLAB 性能加速器的作用，在 MATLAB 中创建例子 4-12 所示的 M 语言代码。

例子 4-12　性能加速器说明——jitaccel.m。

```
tic
A = rand(500000,1);
B = zeros(500000,1);
B(1) = A(1);
for i = 2:500000
   B(i) = B(i- 1) + A(i);
end
toc
```

例子 4-12 中的代码实现了 MATLAB 内建函数——cumsum 函数的功能，将该文件分别在拥有加速器功能的 MATLAB 6.5 和不具备加速器的 MATLAB 6.1 中运行，比较不同的执行时间。

在 MATLAB 6.1 中：

```
>> jitaccel
elapsed_time =
     5.2780
```

在 MATLAB 6.5 中：

```
>> jitaccel
elapsed_time =
     0.0900
```

若直接使用内建函数，则运行的效果为

```
>> tic;B = cumsum(A,1);toc
elapsed_time =
    0.0200
```

通过比较，可以看出 MATLAB 性能加速器明显提高了程序的执行效率，在有些情况下，加速器使执行速度提高 10～100 倍。

注意：

例子 4-12 代码运算得到的时间和具体运行 MATLAB 的计算机的性能有关，不同的计算机环境得到的计算结果不尽相同。另外，MATLAB 性能加速器在 MATLAB 6.5 中的默认设置为开启(On)状态。

4.3 脚 本 文 件

脚本文件是最简单的一种 M 语言文件，在本章前面章节的例子中都使用了脚本文件。所谓脚本文件，就是由一系列的 MATLAB 指令和命令组成的纯文本格式的 M 文件，执行脚本文件时，文件中的指令或者命令按照出现在脚本文件中的顺序依次执行。脚本文件没有输入参数，也没有输出参数，执行起来就像早期的 DOS 操作系统的批处理文件一样，而脚本文件处理的数据或者变量必须在 MATLAB 的公共工作空间中。

例子 4-13 脚本文件示例。

```
% 注释行
%  M 脚本文件示例
% "flower petal"
% 以下为代码行
% 计算
theta = -pi:0.01:pi;
rho(1,:) = 2*sin(5*theta).^2;
rho(2,:) = cos(10*theta).^3;
rho(3,:) = sin(theta).^2;
rho(4,:) = 5*cos(3.5*theta).^3;
for k = 1:4
    % 图形输出
    subplot(2,2,k)
    polar(theta,rho(k,:))
end
disp('程序运行结束!')
```

在 MATLAB 命令行中运行该脚本文件：

```
>> script_example
程序运行结束!
```

MATLAB 会出现相应的图形窗体，如图 4-2 所示。

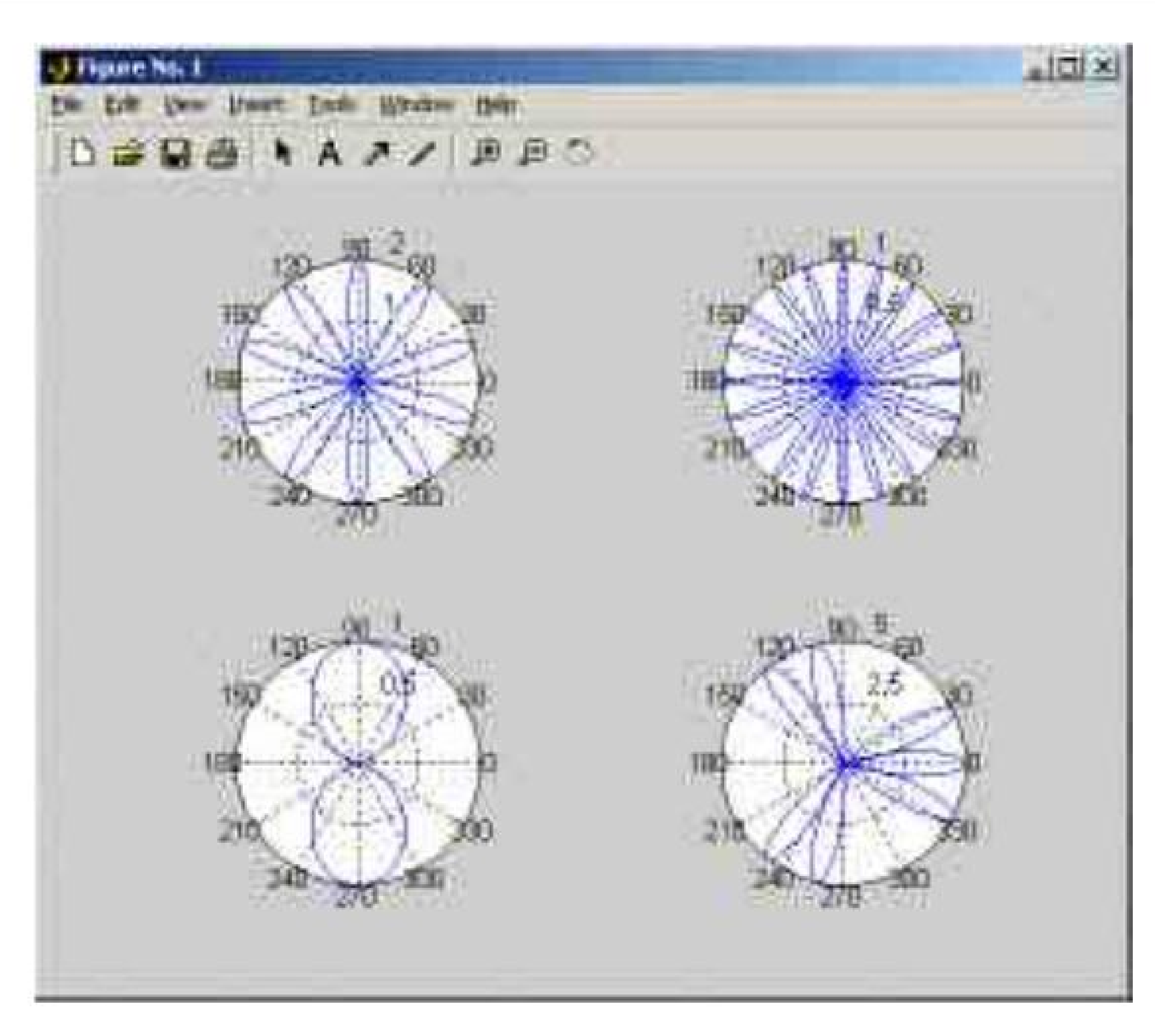


图 4-2　例子 4-13 脚本文件的运行结果

仔细察看例子 4-13 的脚本文件，在脚本文件中，主要由注释行和代码行组成。M 文件的注释行需要使用%定义符，在%之后的所有文本都认为是注释文本，不过，M 文件的注释定义符仅能影响一行代码，类似于 C++语言中的“//”。然而在 M 语言中，没有类似 C 语言的注释定义符“/*”和“*/”，所以无法一次定义多行注释。给程序添加适当的注释是良好的编程习惯，希望读者能够在日常编程中多多使用。

脚本文件中的代码行都是一些简单的 MATLAB 指令或者命令，这些命令可以用来完成相应的计算处理数据、绘制图形结果的操作，也可以在脚本文件中调用其他的函数完成复杂的数学运算，在例子 4-13 中就完成了这些工作。另外，在 MATLAB 中还有一些指令用来处理程序和用户之间的交互，在表 4-4 中进行了总结。

表 **4-4**　脚本文件中常用的 **MATLAB** 指令

指　令	说　　明
pause	暂停当前 M 文件的运行，按任意键继续
input	等待用户输入
keyboard	暂停当前 M 文件的运行，并将程序控制权交还给 MATLAB 命令行，这时可以正常使用命令行，直到键入“return”并按回车键后，M 文件才继续运行
return	返回当前的函数或者命令行

MATLAB 一般使用脚本文件作为某种批处理文件，其中，有两个批处理文件经常被 MATLAB 自动调用，这两个脚本文件分别为 startup.m 和 finish.m。

startup.m 文件在 MATLAB 启动时自动被执行，用户可以自己创建并定义编写该文件，例如在文件中添加物理常量的定义、系统变量的设置或者 MATLAB 搜索路径的设置。当用户安装 MATLAB 之后，在<MATLABROOT>\toolbox\local 路径下有一个 M 文件，名为 Starupsav.m，该文件可以看作是 startup.m 文件的模板，可以修改该文件，然后将其以文件

名 startup.m 的形式保存在<MATLABROOT>\toolbox\local 路径下。

与 startup.m 文件相对应的是 finish.m 文件，该文件在 MATLAB 退出时自动执行，用户可以自己创建并定义编写该文件，例如在文件中添加保存数据等指令，这样可以将每次退出前的工作结果进行保留。同样，在<MATLABROOT>\toolbox\local 路径下有两个文件，分别为 finishsav.m 和 finishdlg.m，这两个文件可以用来作为 finish.m 文件的模板，相关的具体内容介绍请读者自己察看相应的文件和帮助文档。

4.4 函数文件

函数文件是 M 文件最重要的组成部分，M 语言函数文件能够接受用户的输入参数，进行计算，并将计算结果作为函数的返回值返回给调用者。在 MATLAB 中具有不同类型的函数，分别为内建函数、系统 M 函数、系统 MEX 函数文件、用户自定义 MEX 函数文件和用户自定义的 M 文件。其中，内建函数(build-in function)是 MATLAB 基本内核提供的函数，例如三角函数、矩阵运算等函数，这些函数无法察看相应的代码，只能直接使用，所以又称为 MATLAB 核心函数。有关系统的 MEX 函数文件和用户自定义的 MEX 函数文件的内容超出了本书的讨论范围，感兴趣的读者可以阅读《MATLAB 外部接口编程》一书。系统 M 函数是由 MATLAB 提供的 M 语言函数文件，这些函数文件构成了 MATLAB 强大的扩展功能，例如 MATLAB 工具箱中包含的 M 函数文件。而用户自定义的 M 函数文件是由用户自己利用 M 语言编写的文件。在本小节将详细讨论编写 M 语言函数文件的方法。

4.4.1 基本结构

M 函数文件和脚本文件不同，函数文件不仅有自己特殊的文件格式，不同的函数还分别具有自己的工作空间。同其他高级语言类似，M 函数文件也有局部变量和全局变量。读者首先需要了解的是函数文件的基本结构，参见例子 4-14。

例子 **4-14** 函数文件示例——average.m。

```
function y = average(x)
% AVERAGE 求向量元素的均值
% 语法：
% Y = average(X)
% 其中，X 是向量,Y 为计算得到向量元素的均值
% 若输入参数为非向量则出错

% 代码行
[m,n] = size(x);
% 判断输入参数是否为向量
if (~((m == 1) | (n == 1)) | (m == 1 & n == 1))
    % 若输入参数不是向量，则出错
    error('Input must be a vector')
```

```
014    end
015    % 计算向量元素的均值
016    y = sum(x)/length(x);
```

在 MATLAB 命令行中，键入下面的指令运行例子 4-14 的代码：

```
>> z = 1:99;
>> y = average(z)
y =
    50
```

M 语言函数文件具有下面的不同部分：

- 函数定义行。
- 在线帮助。
- 注释行。
- M 语言代码。

下面结合例子 4-14 分别说明这些部分的构成。

函数定义行，例子 4-14 的函数定义行为代码的 001 行：

```
001    function y = average(x)
```

这一行代码中包括关键字 function、函数输出参数 y、函数的名称 average 和函数的输入参数 x。需要读者注意的是函数的名称，函数的名称定义要求必须以字符开头，后面可以用字符、数字和下划线的组合构成函数名称。MATLAB 对函数名称的长度有限定，读者可以在自己的 MATLAB 中，通过执行 namelengthmax 函数获取相应的数值。假设该函数返回的数值为 N，若函数的名称长度超过了 N，则 MATLAB 使用函数名称的前 N 个字符作为函数名称。

一般推荐将函数名称用小写的英文字符表示，同时函数的 M 文件名称最好和函数名称保持一致，若文件名称和函数名称不一致，则调用函数的时候需要使用文件名称而非函数名称。

M 函数文件的在线帮助为紧随在函数定义行的注释行。在例子 4-14 中，average 函数的在线帮助为 002～006 行的注释行。若在 MATLAB 命令行中键入下面的指令：

```
>> help average
```

在 MATLAB 的命令窗口中就会出现：

```
AVERAGE 求向量元素的均值
语法：
  Y = average(X)
  其中，X 是向量，Y 为计算得到向量元素的均值
  若输入参数为非向量则出错
```

其中，在线帮助中比较重要而且特殊的是在线帮助的第一行，在 MATLAB 中将这行注释称为 H1 帮助行，它是在线帮助的第一行，若使用 lookfor 函数查询函数时，仅查询并显示函数的 H1 帮助行，例如，在 MATLAB 命令行中键入下面的指令：

```
>> lookfor average
```

在 MATLAB 的命令窗口中就会出现：

AVERAGE 求向量元素的均值
MEAN   Average or mean value.
⋮

由于 H1 帮助行的特殊作用，所以在用户自己定义 M 函数文件时，一定要编写相应的 H1 帮助行，对函数进行简明、扼要的说明或者解释。

例子 4-14 的 008、010、012、015 行代码分别是程序具体的注释行，这些注释行不会显示在在线帮助中，主要原因就是这些注释行没有紧随在 H1 帮助行的后面，其中 008 行的注释与在线帮助之间有一个空行。其实从 008 行开始一直到文件的结尾都是 M 函数文件的代码行，这些代码行需要完成具体的算法，实现用户的具体功能。代码行就是用户开发的算法 M 语言的实现。

注意：

在创建 M 语言函数文件时，函数的名称最好是全部由小写字符组成。另外，在编写 M 语言函数文件时，函数的在线帮助最好使用英语进行书写，因为 MATLAB 本身是一种国际化的工程软件，使用英语开发在线帮助有利于不同国家和地区的工程师彼此进行技术交流。

4.4.2 输入输出参数

M 语言函数文件的输入、输出参数和其他高级语言的输入、输出参数不同，在定义这些输入、输出参数的时候不需要指出变量的类型，因为 MATLAB 默认这些参数都使用双精度类型，这样可以简化程序的编写。而且在定义参数时，也没有确定输入参数的维数或者尺寸，也就是说，直接从参数上无法判断输入来的是标量、向量还是矩阵，只有通过程序内部的具体代码来加以判断。

M 语言的函数文件不仅可以有一个输入参数和一个返回值，还可以为 M 语言函数文件定义多个输入参数和多个输出参数，见例子 4-15。

例子 4-15　多个输入、输出参数的 M 函数。

```
function [avg, stdev, r] = ourstats(x,tol)
% OURSTATS 多输入输出参数示例
% 该函数计算处理矩阵，得到相应的均值、
% 标准差和矩阵的秩
[m,n] = size(x);
if m == 1
      m = n;
end
% Average
avg = sum(x)/m;
% Sandad deviation
stdev = sqrt(sum(x.^2)/m - avg.^2);
% Rank
s = svd(x);
r = sum(s > tol);
```

运行例子 4-15，在 MATLAB 命令行中，键入下面的指令：

```
>> A = [ 1 2 3; 4 5 6]
A =
     1     2     3
     4     5     6
>> [a,s,r] = ourstats(A,0.1)
a =
    2.5000    3.5000    4.5000
s =
    1.5000    1.5000    1.5000
r =
     2
>>  ourstats(A,0.1)
ans =
    2.5000    3.5000    4.5000
>> [a,s] = ourstats(A,0.1)
a =
    2.5000    3.5000    4.5000
s =
    1.5000    1.5000    1.5000
```

例子 4-15 的 M 代码具有两个输入参数、三个输出参数，所以在使用该函数的时候，需要将必要的输入、输出参数写明。注意调用该函数时的语法，将输出参数依次写在一个向量中，若输出参数的个数与函数定义的输出参数个数不一致，则在例子 4-15 中，将计算得到的前几个输出参数作为返回值，个数等于用户指定的输出参数个数。计算的结果依次赋值给不同的变量。

在使用多个输入、输出参数的时候，往往需要判断用户写明的输入、输出参数的个数，若个数与函数定义不符合的时候，将给出错误或者警告信息，这个时候，需要使用函数 nargin 和 nargout 来获取函数的输入、输出参数个数，见例子 4-16。

例子 4-16　nargin 和 nargout 示例。

```
function c = testarg(a,b)
%TESTARG 检测输入输出参数个数
% 该函数根据不同的输入输出参数个
% 数进行相应的操作
if (nargout ~= 1)
      disp('使用该函数必须指定一个输出参数!');
      return
end
switch nargin
    case 0
```

```
            disp('使用该函数至少需要一个输入参数!');
            c = [];
            return
        case 1
            c = a.^2;
        case 2
            c = a+b;
    end
```

运行例子 4-16，在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = [1 2 3];
>> B = [ 2 3 5];
>> testarg(A,B)
使用该函数必须指定一个输出参数!
>> C = testarg
使用该函数至少需要一个输入参数!
C =
     [ ]
>> C = testarg(A)
C =
     1     4     9
>> C = testarg(A,B)
C =
     3     5     8
>> C = testarg(A,B,C)
??? Error using ==> testarg
Too many input arguments.
```

运行例子 4-16 的代码时，使用不同的输入、输出参数，函数本身和 MATLAB 系统将自动检测参数的个数，在最后一次调用时，由于使用的输入参数个数超过了函数定义的个数，所以 MATLAB 给出了错误信息。

MATLAB 的 M 函数文件还可以具有个数不确定的输入、输出参数，也就是说，在定义 M 函数文件的时候，不指明输入、输出参数的个数，而是在程序中通过编写程序完成具体参数的确定，完成该功能主要依靠 varargin 和 varargout 函数。

当函数的定义具有以下形式的时候

```
function    y = function_name(varargin)
```

函数 function_name 可以接受任意个数的输入参数；而当函数具有下面的形式时

```
function varargout = function_name(n)
```

函数 function_name 可以输出任意个数的输出参数。

可以将 varargin 函数和 varargout 函数结合在同一个 M 文件函数中使用。

使用 varargin 函数接受 M 函数参数时，将所有的用户输入参数构建成为一个元胞数组，该元胞数组中的元素为用户输入的参数，见例子 4-17。

例子 4-17 不确定的输入参数个数。

```
function y = varargin_example(varargin)
%VARARGIN_EXAMPLE 不确定输入参数例子
str = sprintf('输入参数的个数 := %d',length(varargin));
disp(str);
y = 0;
% varargin 的类型
class(varargin)
for i = 1:length(varargin)
    %varargin 为元胞数组
    if(isnumeric(varargin{i}))
        % 将每个为数值数组的输入参数
        % 的第一个元素累加求和
        y = y + varargin{i}(1);
    end
end
```

运行例子 4-17，在 MATLAB 命令行中键入下面的指令：

```
>> varargin_example
输入参数的个数 := 0
ans =
cell
ans =
     0
>> varargin_example('a',[1 2 3],3,rand(2,5))
输入参数的个数 := 4
ans =
cell
ans =
    4.0153
```

通过例子 4-17 的运行结果和例子的代码可以看出 varargin 函数的使用方法。它相当于在 MATLAB 的函数入口处开辟了足够大的空间，用于接受各种用户的输入。在使用这个函数的时候，需要在程序中判别函数输入参数的类别，并且从元胞数组中正确提取变量，在 MATLAB 中将这一过程称之为 unpacking。

与之相对应的，将函数的输入参数传递给 varargout 函数被称之为 packing，在这一过程中，需要将所有必要的输出参数传递给 varargout 元胞数组，在传递参数的时候，还需要注意参数的顺序，在例子 4-18 中介绍了这一过程。

例子 **4-18** 不确定的输出参数。

```
001 function varargout=varargout_example(varargin)
002 %VARARGOUT_EXAMPLE 不确定个数的输出参数
003
004 % 判断输出参数的个数
005 % 下面注释行中的代码执行有错误
006 % str = sprintf('输出参数的个数:=%d',length(varargout));
007 % 必须使用 nargout
008 str = sprintf('输出参数的个数 :=%d',nargout);
009 disp(str);
010 if(nargout <= nargin)
011     for k=1:nargout
012         varargout{k} = varargin{nargin-k+1};
013     end
014 end
```

运行例子 4-18 的代码，在 MATLAB 命令行中，键入下面的指令：

```
>> [a b] =   varargout_example(1,2,3,4)
输出参数的个数 :=2
a =
     4
b =
     3
```

例子 4-18 的第 006 行代码，若将注释行符号“%”删除，则程序运行会出现错误。在程序中，若需要判断输出参数的个数不能使用 length 函数，而需要使用 nargout 函数。另外，在操作输出参数时，需要判断输出参数的个数，根据输出参数的个数完成相应的操作。

在使用不确定的输入、输出参数时，还可以像下面的代码行一样使用这两个参数：

```
function [out1,out2] = example1(a,b,varargin)
function [i,j,varargout] = example2(x1,y1,x2,y2,flag)
```

若使用 varargout 和 varargin 参数，除了必须给定的参数以外，其余的参数是任意数量可变的，具体的操作参阅例子 4-19。

例子 4-19 可变的输入、输出参数。

```
function [x,y,varargout] = vararginout(a,b,c,d,varargin)
%VARARGINOUT 可变的输入输出参数
str = sprintf('输入参数的个数:=%d',nargin);
disp(str);
str = sprintf('输出参数的个数:=%d',nargout);
disp(str);
if(nargin <=4)
    error('输入参数必须多于 4 个');
```

```
009     end
010     % 处理输入输出参数
011     x = a+b+c+d;
012     y = a-b+c-d;
013     if(nargout >2 && nargin>4)
014         for(i =1:nargout-2)
015             % 这里也许会出错，小心！
016             varargout{i} = varargin{end-i-1};
017         end
018     end
```

运行例子 4-19 的代码，在 MATLAB 命令行中，键入下面的指令：

```
>> vararginout(1, 2, 3, 4, 5)
输入参数的个数: =5
输出参数的个数: =0
ans =
    10
>> [a,b,c]=vararginout(1, 2, 3, 4, 5, 6, 7, 8)
输入参数的个数: =8
输出参数的个数: =3
a =
    10
b =
    -2
c =
     6
```

这里就不再对例子 4-19 的代码进行详细的解释了，请读者自行查阅，不过在代码的 015 行，运行该行代码时有可能出现错误，有兴趣的读者可以仔细察看，为什么此行代码可能会出错，并对程序进行修改，来保证程序不出错。

4.4.3 子函数和私有函数

同一个 M 函数文件中可以包含多个函数。如果在同一个 M 函数文件中包含了多个函数，那么将出现在文件中的第一个 M 函数称为主函数(primary function)，其余的函数称为子函数(subfunction)。M 函数文件的名称一般与主函数的名称保持一致，其他函数都必须按照函数的基本结构来书写，每一个函数的开始都是函数定义行，函数的结尾是另一个函数的定义行的开始或者整个 M 文件的结尾(最后一个子函数的结尾就是文件结束符)。不过，子函数不像主函数，一般子函数没有在线帮助，子函数的作用范围有限，它只能被那些在定义子函数的 M 文件中定义的函数(包括主函数和其他子函数)调用，不能被其他 M 文件定义的函数调用。

例子 4-20　子函数应用例子。

```
001 function [avg,med] = newstats(u) % 主函数
002 % NEWSTATS 计算均值和中间值
003 n = length(u);
004 avg = mean(u,n);                  % 调用子函数
005 med = median(u,n);                % 调用子函数
006
007 function a = mean(v,n)            % 子函数
008 % 计算平均值
009 a = sum(v)/n;
010
011 function m = median(v,n)          % 子函数
012 % 计算中间值
013 w = sort(v);
014 if rem(n,2) == 1
015     m = w((n+1)/2);
016 else
017     m = (w(n/2)+w(n/2+1))/2;
018 end
```

运行例子 4-20，在 MATALB 命令行窗口中，键入下面的指令：

```
>> x = 1:11;
>> [mean,mid] = newstats(x)
mean =
     6
mid =
     6
```

在例子 4-20 的代码中，分别在 007 行和 011 行定义了两个子函数 mean 和 median，这两个子函数分别在主函数的 004 行和 005 行被调用。注意，在不同的函数之间传递变量是通过函数的参数形式来完成的。

在 MATLAB 中有一类函数被称为私有函数，这类函数被放置在名称为 private 的子目录中。每一个函数文件都是标准的 M 语言函数文件，没有特殊的关键字。但是，这些函数仅能被那些位于 private 子目录的上一层目录中的函数调用。例如，假设在 MATLAB 的搜索路径中包含路径\ProjectA ，那么所有位于 \ProjectA\private 路径下的函数，只能在其上一层路径 \ProjectA 中的函数文件中调用。由于私有函数作用范围的特殊性，不同父路径下的私有函数可以使用相同的函数名。由于 MATLAB 搜索函数时优先搜索私有函数，所以如果同时存在私有函数名 func1.m 和非私有函数名 func1.m，则私有函数 func1.m 被优先执行。

创建私有函数的方法非常简单，只要将那些需要设置为私有的函数都拷贝到一个 private 子目录中，则这些函数就能被那些位于父层目录中的 M 函数调用了。

在表 4-4 中总结了子函数和私有函数的区别。

表 **4-4**　私有函数和子函数比较

函数类型	子　函　数	私 有 函 数
作用范围	同一个 M 函数文件内	在上层路径中的函数文件内
结构	保存在同一个 M 语言函数文件中，在 M 语言文件可以不包含任何子函数	保存在子目录 private 下

例子 4-21　私有函数的例子。

创建一个新的函数文件，代码如下：

```
function x = pmean(v,n)
%MEAN 私有函数例子
% 将该函数文件保存在 pivae 子目录中，
% 则该函数仅能在上层目录的函数文件
% 中调用
disp('私有函数 mean');
x =  sum(v)/n;
```

使用例子 4-20 函数文件调用该文件，则在 newstats.m 文件所在的路径下创建一个名为 private 的子目录，然后将 pmean.m 文件拷贝到该路径中，如图 4-3 所示。

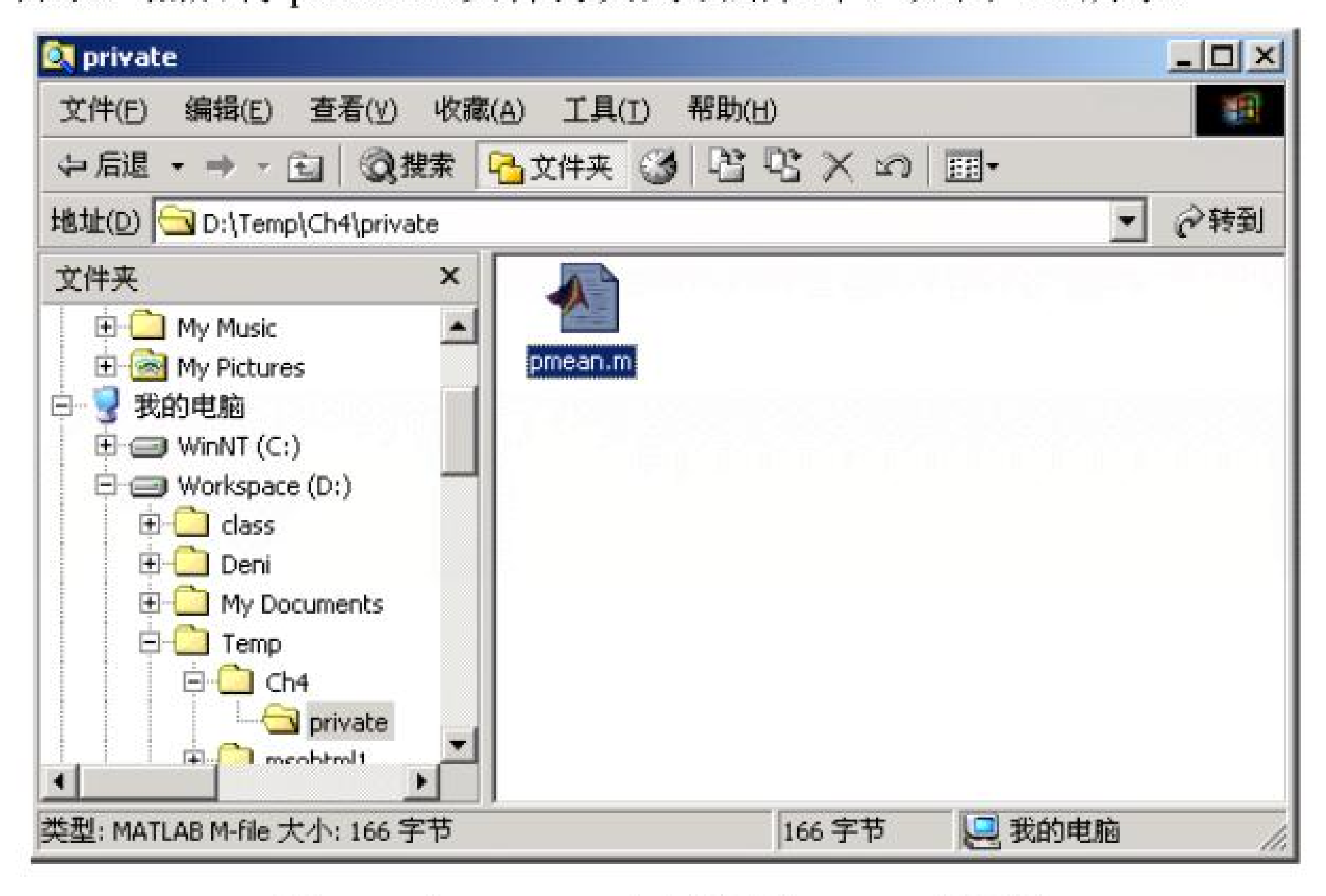


图 4-3　将 pmean.m 文件保存在 private 目录下

接着，修改 newstats 函数，并将其另存为 newstats1.m。

```
function [avg,med] = newstats1(u) % 主函数
% NEWSTATS 计算均值和中间值
n = length(u);
avg = mean(u,n);                  % 调用子函数
avg1 = pmean(u,n)                 % 调用私有函数
med = median(u,n);                % 调用子函数

```

```
008    function a = mean(v,n)            % 子函数
009    % 计算平均值
010    disp('子函数 mean');
011    a = sum(v)/n;
012    ……
```

然后在 MATLAB 命令行中，执行 newstats1.m 函数：

```
>> newstats1(1:10);
子函数 mean
avg =
     5.5000
私有函数 mean
avg1 =
     5.5000
```

注意 newstats1.m 文件的 005 行，这里调用了私有函数。其实私有函数与子函数的使用过程并没有很大的区别，不过两种函数的作用范围却有很大的差别，请读者注意。

4.4.4 局部变量和全局变量

同 C 语言类似，在 M 语言函数中也存在局部变量和全局变量。所谓局部变量，就是那些在 M 函数内部声明并使用的变量。这些变量仅能在函数调用执行期间被使用，一旦函数结束运行，则这些变量占用的内存空间将自动被释放，变量的数值也就不存在了。这是由于 MATLAB 的解释器在解释执行函数的时候，为不同的函数创建不同的工作空间，函数彼此的工作空间相互独立，一旦函数执行完毕，则函数的工作空间就不存在了。

在本章前面的例子中，每个例子的函数内部声明使用的变量都是局部变量，所以函数执行完毕后，MATLAB 的基本工作空间中没有这些变量存在，参见例子 4-22。

例子 4-22 局部变量的例子。

```
function local
%LOCAL 察看局部变量的例子
x = rand(2,2);
y = zeros(2,2);
z = '函数中的变量';
u = {x,y,z};
disp(z)
whos
```

运行例子 4-22，在 MATLAB 命令行中，键入下面的指令：

```
>> local
函数中的变量
  Name        Size                  Bytes  Class
  u           1x3                     256  cell array
```

```
  x          2x2                    32   double array
  y          2x2                    32   double array
  z          1x6                    12   char array
Grand total is 31 elements using 332 bytes
>> whos
```

通过运行 local 函数可以看到，所有在函数中创建的变量在函数运行结束后就不存在了。也就是说，局部变量的生存周期仅在函数的活动期间内。

与局部变量相对应的就是全局变量。MATLAB 将全局变量保存在特殊的工作空间进行统一维护、管理，而将变量声明为全局变量的方法就是在使用变量前，用关键字 global 声明，例如声明全局变量 gXY：

```
>> global gXY
>> whos
  Name       Size                  Bytes   Class
  gXY        0x0                       0   double array (global)
Grand total is 0 elements using 0 bytes
```

需要强调一点，MATLAB 管理、维护全局变量和局部变量使用了不同的工作空间，所以使用 global 关键字创建全局变量的时候有三种情况：

(1) 若声明为全局的变量在当前的工作空间和全局工作空间都不存在，则创建一个新的变量，然后为这个变量赋值为空数组，该变量同时存在于局部工作空间和全局工作空间。

(2) 若声明为全局的变量已经存在于全局工作空间中，则不会在全局工作空间创建新的变量，其数值同时赋值给局部工作空间中的变量。

(3) 若声明为全局的变量存在于局部工作空间中，而全局工作空间不存在，则系统会提示一个警告信息，同时将局部的变量“挪”到全局工作空间中。

例子 4-23　全局变量的例子。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>>% 创建全局变量并赋值
>> global myx
>> myx = 10;
>>% 变量的信息
>> whos
  Name       Size                  Bytes   Class
  myx        1x1                       8   double array (global)
Grand total is 1 element using 8 bytes
>>% 清除变量
>> clear myx
>>% 察看信息
>> whos
>> whos global
  Name       Size                  Bytes   Class
```

```
  myx          1x1                          8   double array (global)
Grand total is 1 element using 8 bytes
>>% 在局部工作空间再次创建变量
>> myx = 23
myx =
    23
>>% 变量的信息
>> whos
  Name         Size                     Bytes   Class
  myx          1x1                          8   double array
Grand total is 1 element using 8 bytes
>>% 将其修改为全局变量(注意警告信息)
>> global myx
Warning: The value of local variables may have been changed to match the
         globals.   Future versions of MATLAB will require that you declare
         a variable to be global before you use that variable.
>>%看看变量的数值
>> myx
myx =
    10
>>% 清除当前的工作空间
>> clear
>> whos global
  Name         Size                     Bytes   Class
  myx          1x1                          8   double array (global)
Grand total is 1 element using 8 bytes
>>% 清除所有的内存空间
>> clear all
>> whos global
```

在例子 4-23 中充分说明了全局变量与局部变量之间的差异，特别要注意局部变量转变为全局变量的过程，如例子所示，这个过程中原来局部变量的数值丢失了，请读者在使用全局变量时务必注意！

注意：

对任何变量操作之前，建议使用 global my_x 命令，这将把全局和本地变量联系起来，不会产生警告信息，减少了变量冲突。

使用全局变量时，需要小心留意，因为全局变量可以在任何的函数中进行读写，这样，可能在比较复杂的程序中查找全局变量错误的时候就非常的麻烦。

在 MATLAB 中还有一类变量被声明为 persistent，本书将其称之为保留变量，这类变量类似于 C 语言函数中被声明为 static 类型的变量。这类变量在函数退出的时候不被释放，当

函数再一次被调用的时候，这些变量保留上次函数退出时的数值。被声明为 persistent 的变量具有以下特征：

- 变量仅能在声明变量的函数内使用，其他函数不能直接使用这些变量。
- 函数执行退出后，MATLAB 不清除这些变量占用的内存。
- 当函数被清除或者重新编辑后，保留的变量被清除。

关于 persistent 关键字和保留变量的使用参见例子 4-24。

例子 4-24　persistent 关键字。

```
function y = persistent_example(x)
%PERSISTENT_EXAMPLE 保留变量使用示例
for i = 1:x
    y = myfun;
end

function y = myfun
% 子函数
% persistent 关键字的使用
persistent count;
% count 记录函数调用的次数
if( isempty(count))
    count = 1;
else
    count = count + 1;
end
str = sprintf('第%d 次调用该函数',count);
disp(str);
y = count;
```

在 MATLAB 命令行窗口中，执行该函数：

```
>> persistent_example(2)
第 1 次调用该函数
第 2 次调用该函数
ans =
     2
>> persistent_example(3)
第 3 次调用该函数
第 4 次调用该函数
第 5 次调用该函数
ans =
     5
```

从例子 4-24 的执行情况可以看出变量 count 记录了函数被调用的次数，如果在 MATLAB 命令行中键入如下指令：

```
>> clear all
```

则所有的变量都会被清除，此时再次执行例子 4-24：

```
>> persistent_example(3)
第 1 次调用该函数
第 2 次调用该函数
第 3 次调用该函数
ans =
     3
```

count 数值又重新计数了。

由于使用全局变量有这样那样的危险性，所以建议读者尽量使用函数参数传递的方式来完成函数之间的数据共享，或者可以使用 persistent 关键字将必要的变量保护起来。另外，isglobal 命令可以用来测试本地变量与全局工作区中的变量是否存在联系，该命令并不能判断全局工作区中是否存在该变量。如果全局工作区中存在某个变量，但与本地工作区中的相应变量没有联系，isglobal 函数返回值为 0 (假)。

4.4.5 函数执行规则

到这里，读者应该能够创建自己的算法函数，并且能够执行任何 M 语言函数了。只要在 MATLAB 的命令行窗口中键入函数的名称，并且提供足够的输入、输出参数就会得到正确的结果。如前文所述，M 语言的函数被 MATLAB 的解释器解释、执行，所以，在本小节中将简要讨论一下解释器解释执行程序的问题。

当用户在 MATLAB 命令行窗口键入一条命令或者执行 M 语言文件中包含的一条语句或者指令时，MATLAB 解释器就负责解析用户的输入，并且给出相应的答案。MATLAB 解释器解析命令按照一定优先级进行：

- 首先判断输入的命令是否为变量。
- 若不是内存中的变量，判断输入的命令是否为 MATLAB 的内建函数。
- 若不是内建函数，则判断输入的命令是否为子函数。
- 若不是子函数，则判断输入的命令是否为私有函数。
- 若不是私有函数，则判断输入的命令是否为 MATLAB 搜索路径中所包含的某个文件或函数。
- 若在同一个路径下发现同名的三种类型的文件 MEX 文件、P 代码文件和 M 代码文件，则优先执行 MEX 文件，其次是 P 代码文件，最后执行的是 M 语言文件。

提示：

若需要了解具体调用的是哪一个对象，则可以使用 which 命令获取相应的信息。

这里需要注意的一点就是，MATLAB 内存中的变量比函数具有较高的优先级，如例子 4-25 所示。

例子 4-25　MATLAB 命令解析的优先级。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> cos = 'This is a string!'
cos =
This is a string!
>> cos(4)
ans =
s
>> which cos
cos is a variable.
>> clear all
>> cos(4)
ans =
       - 0.6536
>> which cos
cos is a built-in function.
```

通过例子 4-25 说明，MATLAB 命令解释器在解释、执行 MATLAB 指令时，将变量置于第一位，所以读者在进行 M 语言编程的时候注意自己定义的变量名称，避免命名冲突。

提示：

P 代码(伪代码)文件是从 M 文件用 pcode 命令生成的。伪代码是经过预编译的，无论何时函数被调用，MATLAB 都能访问的现成代码。因为 P 文件是预编译过的，所以它们的实际内容对用户而言是不可读的，而且在一般情况下它们比相应的 M 文件运行速度快。将 M 语言函数文件转变为 P 代码文件的方法是

```
>> pcode fun1 fun2………
```

MEX 文件是一种特殊的文件格式，主要使用 C 语言进行开发，并且经过编译后在 MATLAB 环境中使用，关于 MEX 文件的详细信息将在《MATLAB 外部接口编程》一书中详细讨论。

请读者一定要牢记函数文件调用的优先级规则，充分利用这些规则可以有效提高 M 文件的执行效率，简化代码的书写。

4.5　M 文件调试

M 语言文件的编辑器——meditor 不仅仅是一个文件编辑器，同时还是一个可视化的调试开发环境。在 M 文件编辑器中可以对 M 脚本文件、函数文件进行调试，以排查程序的错误。M 文件的调试不仅可以在文件编辑器中进行，而且还可以在命令行中结合具体的命令进行，但是过程相对麻烦一些，所以本小节将重点讲述在 M 文件编辑器中进行可视化调试的过程。

一般地来说，应用程序的错误有两类，一类是语法错误，另外一类是运行时的错误。

其中，语法错误包括了词法或者文法的错误，例如函数名称的拼写错误等。而运行时的错误是指那些程序运行过程中得到的结果不是用户需要的情况。但是，由于 M 文件是一种解释型语言，语法错误和运行时的错误都只有在运行过程中才能发现，所以程序的调试往往是在程序无法得到正确结果时进行程序修正的惟一手段。

为了能够有效地处理各种情况，M 语言的断点类型除了类似 C 语言的用户定义的断点外，还有几种自动断点，分别为

- Stop if Error。
- Stop if Warning。
- Stop if NaN or Inf。
- Stop if All Errors。

这些自动断点可以在程序中设置，当程序运行过程中发生了错误或者警告，则程序运行中断，进入调试状态。

为了能够进行可视化的程序调试，M 语言文件编辑器提供了可视化的工具帮助用户进行程序调试，其中比较重要的就是 Breakpoints 菜单，如图 4-4 所示。

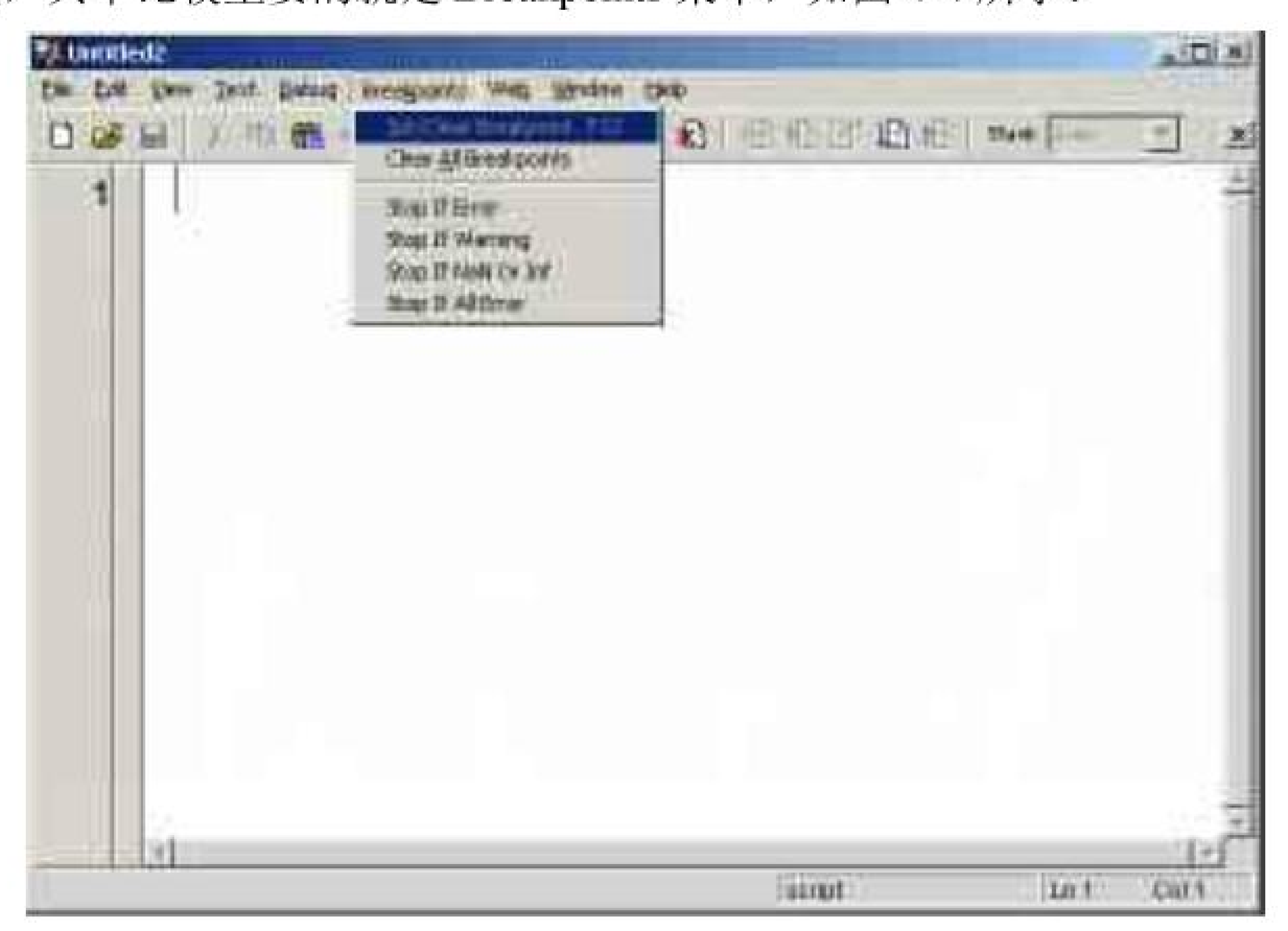


图 4-4　M 文件编辑器的 Breakpoints 菜单

Breakpoints 菜单中的命令用来设置各种断点，只有设置了断点的应用程序才能够进行调试。这里结合具体的例子说明 M 文件的调试过程。

例子 4-26　M 文件调试代码——stats_error.m。

```
function [totalsum,average] = stats_error (input_vector)
% STATS_ERROR - Calculates cumulative total & average
totalsum = sum(input_vector);
average = ourmean(input_vector);

function y = ourmean (x)
```

```
007     % OURMEAN - Calculates average
008     [m,n] = size(x);
009     if m == 1
010         m = n;
011     end
012     y = sum(input_vector)/m;
```

首先在 MATLAB 环境中启动 M 文件编辑器，然后选择 M 文件编辑器中“Breakpoints”菜单下的“Stop if Error”命令。注意，这时不一定需要将 stats_error.m 文件在文件编辑器中打开。

然后，在 MATLAB 命令行窗口中键入下面的指令：

```
>> [sum avg] = stats_error(rand(1,50))
??? Undefined function or variable 'input_vector'.


Error in ==> D:\TEMP\ch4\stats_error.m (ourmean)
On line 12    ==> y = sum(input_vector)/m;
```

MATLAB 首先提示程序运行有错误，并且指出错误发生的地点，接着就会自动地在 M 文件编辑器中加载运行的 M 文件，如图 4-5 所示。

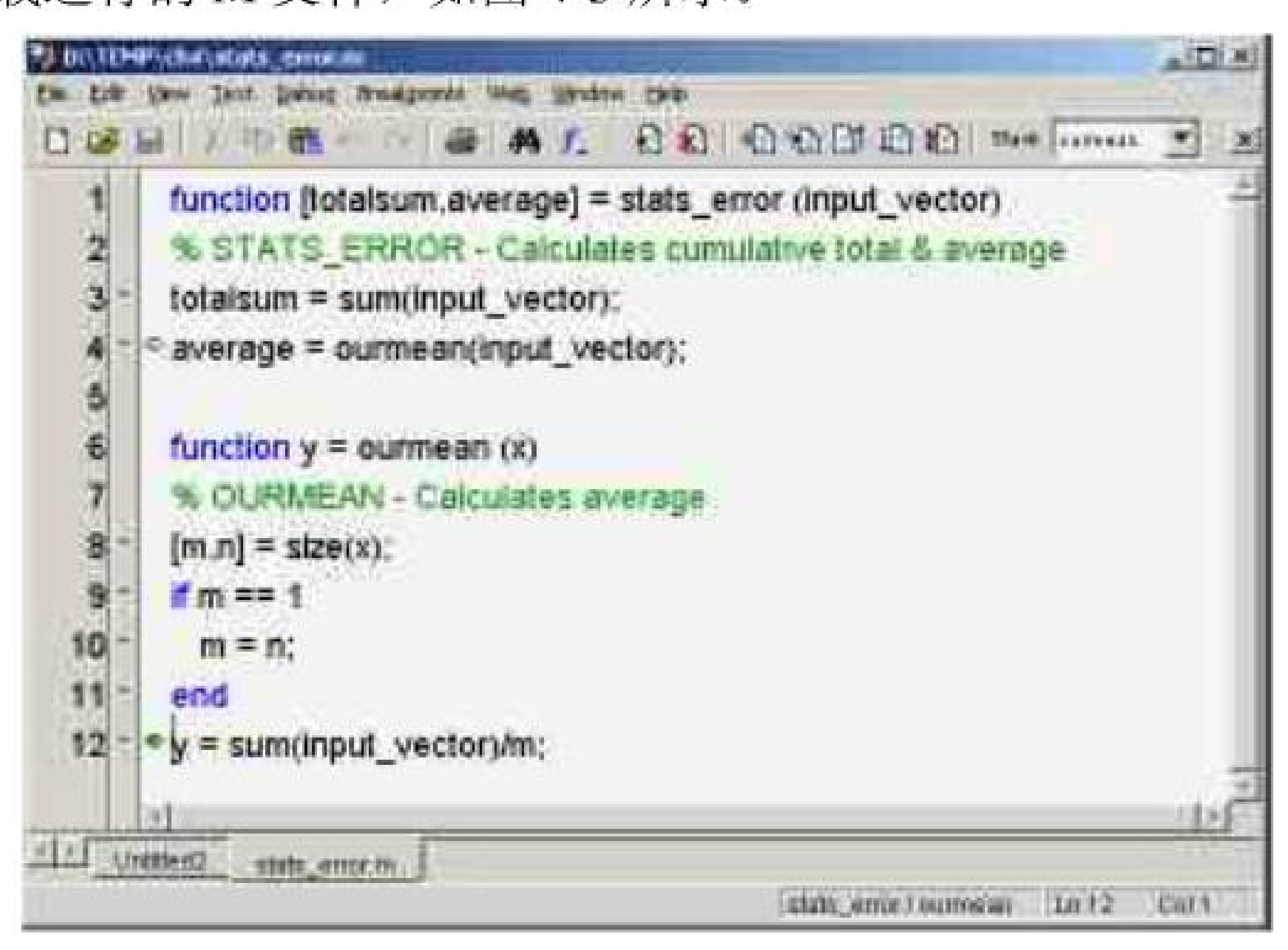


图 4-5　处于调试状态的 M 文件编辑器

在 M 文件编辑器中，第 12 行代码前有绿色的箭头，表示当前程序运行在此处中断。通过用户界面中的 Stack 下拉框可以察看当前应用程序使用堆栈的状态，如本例子中 Stack 下拉框中包含如下的内容：ourmean、stats_error 和 Base，由下至上，分别为调用者和被调用者之间的关系，同时也显示了当前的工作空间。另外，部分按钮从编辑状态进入调试状态，如图 4-6 所示。

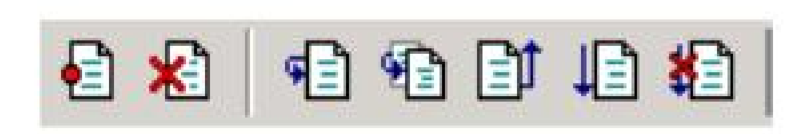


图 4-6　调试程序的按钮

这些按钮分别执行增加断点、清除断点、单步执行等调试程序的功能。将鼠标光标移动到按钮处并保持几秒钟，MATLAB 的文件编辑能够给出相应的提示。

此时，MATLAB 命令行窗口也处于调试状态，在这种状态下命令行提示符为“K>>”，在该命令行提示符中可以任意键入 MATLAB 指令进行运算和处理，不过需要注意，此时的工作空间是函数正在应用的空间，若在命令行窗口中键入的指令影响了工作空间中的变量，则可以直接影响程序运行的结果。

例如，在当前的提示符“K>>”键入下面的指令：

```
K>> whos
  Name      Size                    Bytes  Class
  m         1x1                         8  double array
  n         1x1                         8  double array
  x         1x50                      400  double array
Grand total is 52 elements using 416 bytes
```

可以看到，当前的工作空间下没有变量名 input_vector，这也是该程序执行出错的原因，将程序中第 12 行的 input_vector 修改成为 x 就能得到正确的答案了。

MATLAB 可视化程序调试功能相对于 Visual C++的可视化调试功能弱了一些，但是，在调试程序的过程中通过 MATLAB 命令行窗口的配合，充分利用 MATLAB 命令行窗口“演算纸”的功能，能够非常方便地调试 M 语言应用程序。

另外，MATLAB 也提供了一些指令用于进行 M 文件的调试，在表 4-5 中对这些命令进行了总结。

表 **4-5** 应用与调试 **M** 文件的指令

指　令	说　　明
dbclear	清除已经设置好的断点
dbcont	继续执行，等同于工具栏中的按钮
dbdown/dbup	修改当前工作空间的上、下文关系
dbquit	退出调试状态
dbstack	显示当前堆栈的状态
dbstatus	显示所有的已经设置的断点
dbstep	执行应用程序的一行或者多行代码
dbstop	设置断点
dbtype	显示 M 文件代码和相应的行号

表 4-5 中指令的具体使用方法请读者查阅在线帮助或者 MATLAB 的帮助文档，在本书中就不再赘述了。其中，比较常用的指令是 dbquit，在可视化调试过程中，往往会出现没有退出调试状态就关闭了 M 文件编辑器的情况，这时可以在“K>>”提示符下键入该指令退出调试状态。另外，在 startup.m 文件中利用 dbstop 指令预先设置自动断点为有效，这样就不必每次在调试应用程序前设置自动断点了。

4.6 M 文件性能分析

前面的小结介绍了提高 M 语言代码性能的方法，但是在进行实际编程的时候如何得知

自己编写程序的执行效率呢？使用 toc 和 tic 函数仅仅能够获得整个程序执行过程中消耗的时间，很多时候需要了解具体哪一部分代码占用了最多的执行时间，这个时候就需要使用 M 文件性能分析器了。利用 M 文件性能分析器可以有效地帮助用户了解具体哪一行或哪一段代码占用了最多的计算时间，让程序开发人员能够尽快地把握住程序运行的瓶颈。

使用 M 文件性能分析器有两种方法，其中一种方法是通过命令行，另外一种方法是通过图形用户界面。本小节分别介绍这两种使用 M 文件性能分析器的方法。

本小节用来分析的程序为例子 4-10 向量化运算——array_vs_loops.m 的代码。

使用性能分析器的图形用户界面，通过执行 MATLAB 的“Start”菜单中“MATLAB”子菜单下的“Profiler”命令，得到性能分析器的用户界面，如图 4-7 所示。



图 4-7　性能分析器的图形用户界面

提示：

在 MATLAB 命令行窗口中键入指令：

```
>> profview
```

同样可以打开 M 文件性能分析器的图形用户界面。

然后，在“Run this code”下拉框中，键入需要进行性能分析的 MATLAB 指令。

注意：

这里需要键入执行 M 文件的指令，而不是简单地键入文件名称。这里的指令等同于在 MATLAB 命令行窗口下键入的执行应用程序的指令。

键入 array_vs_loop，然后单击“Start Profiling”按钮，则性能分析器通过运行指定的指令得出程序运行消耗的时间，并给出一个简要的总结，如图 4-8 所示。

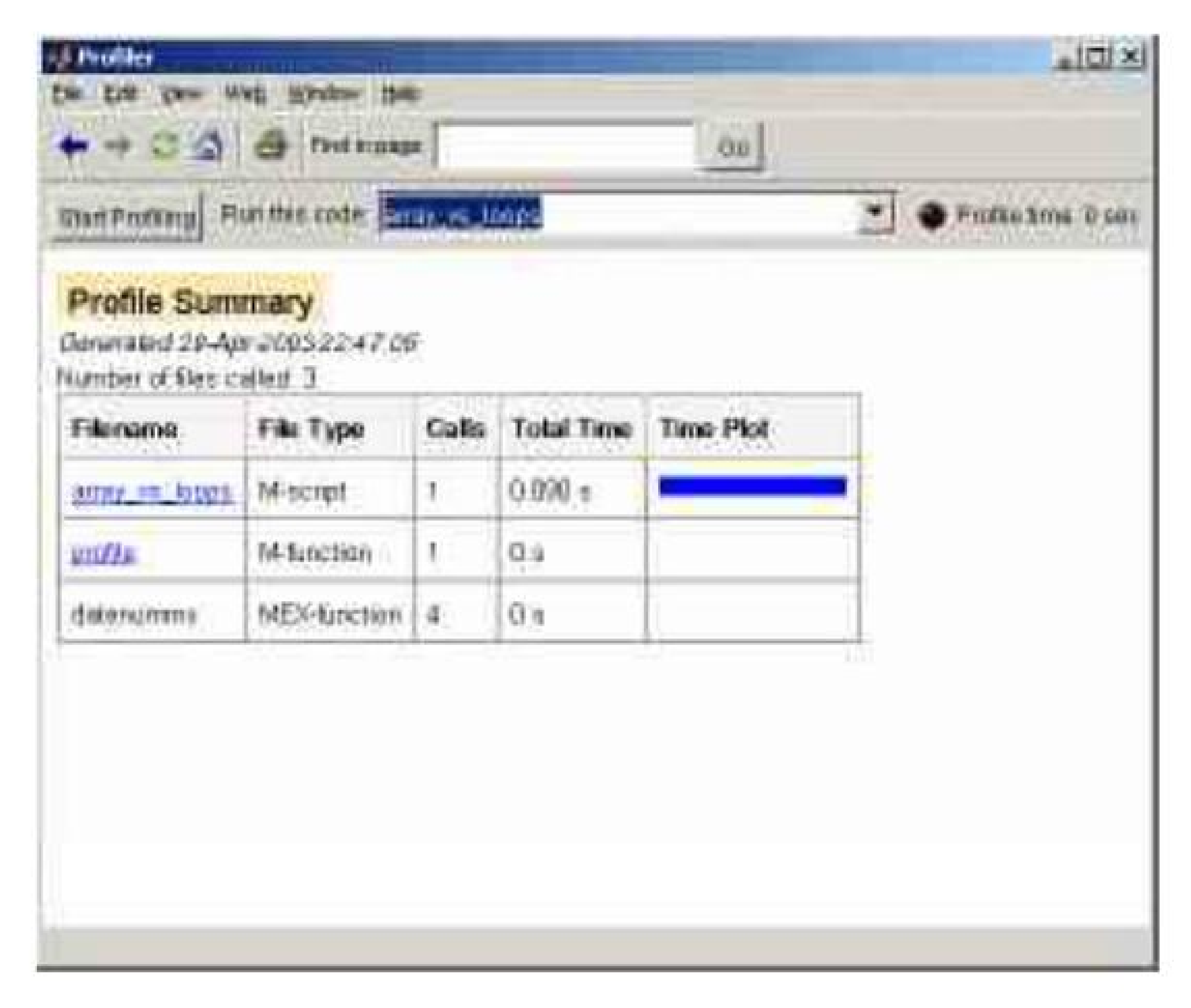


图 4-8 性能分析的总结报告

单击总结报告中的“array_vs_loops”超链接，就能够得到 array_vs_loops 文件运行的详细分析报告，在这个超文本格式的报告中，用不同的色彩区分了执行时间、代码执行的次数等信息，如图 4-9 所示。



图 4-9 性能分析的详细报告——执行次数报告

详细分析报告主要按照不同的项目统计了程序的执行，其中包括运行时间(Time)、语句调用次数(Number of Call)、覆盖率(Coverage)、已加速的代码(Acceleration)。从分析报告中就可以得出占用了较多计算处理时间的代码段或代码行。另外，通过性能分析器的报告也

能看出使用数组运算大大提高了 M 文件的执行性能。读者进行到这里可以单击详细报告上的所有超链接，获取关于 M 文件的详细信息。

这里比较重要的就是察看代码的加速信息了，在 M 语言文件性能分析器中，如果代码的前面具有“x”标识，则说明此行代码没有被 MATLAB 性能加速器加速，用户可以单击“x”标识，则性能分析器将弹出对话框说明此行代码没有被加速的具体原因。用户可以充分利用该对话框的提示信息修改代码，以提高程序的执行效率。

注意：

由于系统的限制，截至笔者截稿前的 MATLAB 最新版本，图形用户界面的性能分析器都不能处理任何包含中文字符的 M 文件代码。

另外一种进行性能分析的方法是直接利用命令行的方式进行，这里主要用到一个函数——profile，该函数的主要使用方法为

```
profile keywords
```

其中，根据不同的关键字执行不同的功能，而使用的过程大体分为三个步骤：

首先，在 MATLAB 命令行中键入下面的指令：

```
>> profile on
```

其中，关键字 on 的作用是开启性能分析器，并且将前面的统计结果清除。

注意：

当性能分析器处于开启状态时，MATLAB 的状态栏显示“Profile on”。

其次，运行需要分析的 M 文件，例如，本小节使用的 array_vs_loops.m 文件。运行文件的方法和正常运行 M 文件的方法一致：

```
>> array_vs_loops
```

第三步，也就是最后一步，在 MATLAB 命令行中键入下面的指令：

```
>> profile report
```

这时 MATLAB 将分析的结果创建为一个超文本格式的文档，该文档就是性能分析的总结文档，如图 4-10 所示。

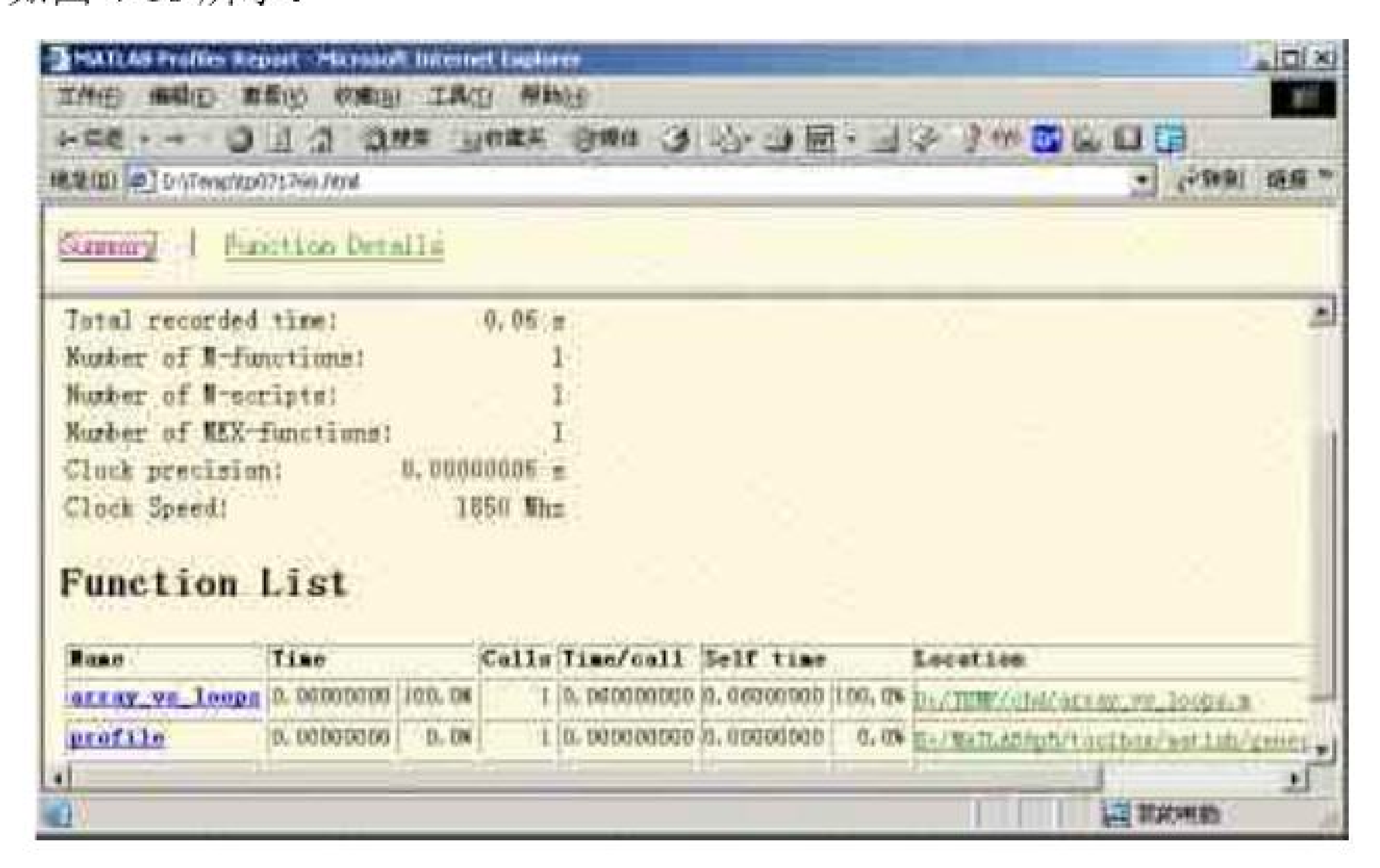


图 4-10　性能分析结果

通过单击相应的超链接可以获取相应文件的分析报告，其主要内容为每一行代码执行占用的时间。和使用图形用户界面的性能分析器相比较，这个分析报告的内容相对简单，只有执行时间的报告，但是能够支持包含中文的语句，如图 4-11 所示。

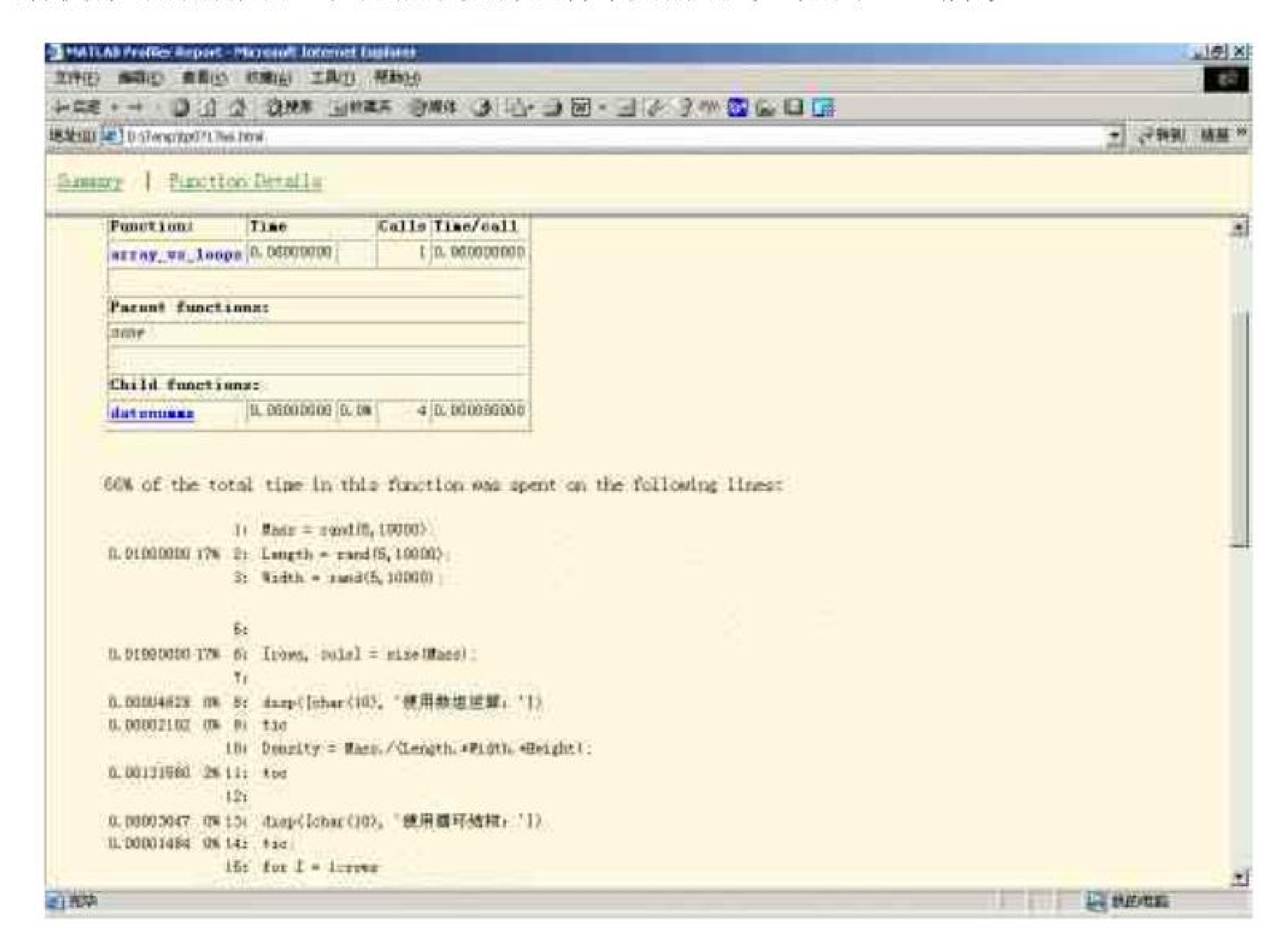


图 4-11 性能分析的详细报告

在表 4-6 中，总结了 profile 函数的使用方法。

表 **4-6 profile** 函数的使用方法

语 法	说 明
profile on	开启性能分析器，同时清除上一次的统计信息
profile off	关闭性能分析器
profile report	关闭性能分析器，同时输出超文本的报告
profile plot	关闭性能分析器，同时以柱状图绘制统计结果
profile resume	重新开启性能分析器，保留前一次的统计结果
profile clear	清除统计结果
profile viewer	打开图形界面的性能分析器
S = profile(‘status’)	将性能分析器的状态保存在结构 S 中
S = profile(‘info’)	关闭性能分析器，同时将分析结果保存在结构 S 中

关于 profile 函数更详细的使用说明请参阅 MATLAB 的帮助文档。

4.7 本 章 小 结

在本章主要讲述了关于如何利用 MATLAB 的编程语言——M 语言进行编程的方方面面。MATLAB 提供了一种高级编程语言——M 语言，这种语言的语法结构与 C 语言非常类似，任何熟悉 C 语言的用户学习使用 M 语言都不会有任何障碍。尽管 M 语言是一种解释性的编程语言，但是随着 MATLAB 版本的不断升级，以及充分利用 MATLAB 提供的各种编程技巧，能够有效提高 M 语言应用程序的执行效率，使 M 语言成为了工程领域中最适合进行算法开发验证的编程语言。

通过本章的学习之后，读者应该能够比较熟练地利用 M 语言实现自己的想法。MATLAB 是灵活、可靠的开发环境，用户不仅可以利用已有的 MATLAB 的功能，而且还能够利用 M 语言丰富 MATLAB 的能力。通过本章的学习，读者还能够充分利用 MATLAB 新版本特性，以及 MATLAB 丰富的编程调试工具——调试器和性能分析器，编写出高效率可靠的应用程序。

本章所涉及的内容仅仅是 M 语言的一部分内容，请读者仔细阅读 MATLAB 的帮助文档，已获得更加全面的信息。

第五章　文 件 I/O

在日常工作中工程师总是在和各种各样的数据打交道，而计算机中，不同的数据往往使用不同的文件格式保存。例如常微分方程数值解通常保存成为文本格式、图像处理的结果是某种图形图像格式文件、语音信号被保存为 WAV 格式或者经过数字压缩得到 MP3 格式、金融工程师将大量的数据保存在 Excel 电子表格中。MATLAB 作为一种科学计算及数据处理的平台，提供了丰富的功能用于数据文件的输入、输出。在本章，将详细讲解在 MATLAB 中输入、输出数据文件的基本方法。

本章讲述的主要内容如下：

- 文件 I/O 的高级例程；
- 文件 I/O 的低级例程；
- 数据文件的导入、导出向导。

5.1　概　　述

MATLAB 提供了丰富的手段进行数据文件的输入、输出。其中，MATLAB 把从磁盘或者剪贴板获取数据到 MATLAB 的工作空间的过程称之为导入(Importing)数据，把数据从 MATLAB 的工作空间中按照一定的格式保存到磁盘的过程称之为导出(Exporting)数据。MATLAB 导入数据支持的文件格式种类繁多，略加分类包含文本格式文件、二进制格式文件以及其他标准格式文件。用户需要根据不同的需要(导入数据还是导出数据)以及文件的格式(文本或二进制)选择不同的文件 I/O 方式。

MATLAB 自己提供一种特殊的数据文件格式——MAT 文件，这种文件是一种二进制格式文件，扩展名为 .mat，它为 MATLAB 提供了跨平台的数据交互能力。这些 *.mat 文件之所以能够独立于各种平台的原因是在文件头带有设备的签名，MATLAB 在载入文件时将检查这个签名，如果发现文件来源不同于当前的系统，则进行必要的转换。目前 MAT 文件的版本为 5，它的文件格式如图 5-1 所示。

一般 MAT 文件分为两个部分：文件头部和数据。其中在文件的头部主要包括一些描述性文字和相应的版本与标识，这部分占用了 128 个字节。此后依次是保存在 MAT 文件中的数据，数据是按照数据类型、数据长度和数据三个部分保存的。

MAT 文件不仅可以被 MATLAB 的函数加载，而且还能被 C 或者 Fortran 语言编写的程序读写，MATLAB 提供了相应的 API 用于这些应用程序的编写。有关 MAT 文件的 C/Fortran 语言 API 参阅《MATLAB 外部接口编程》一书。

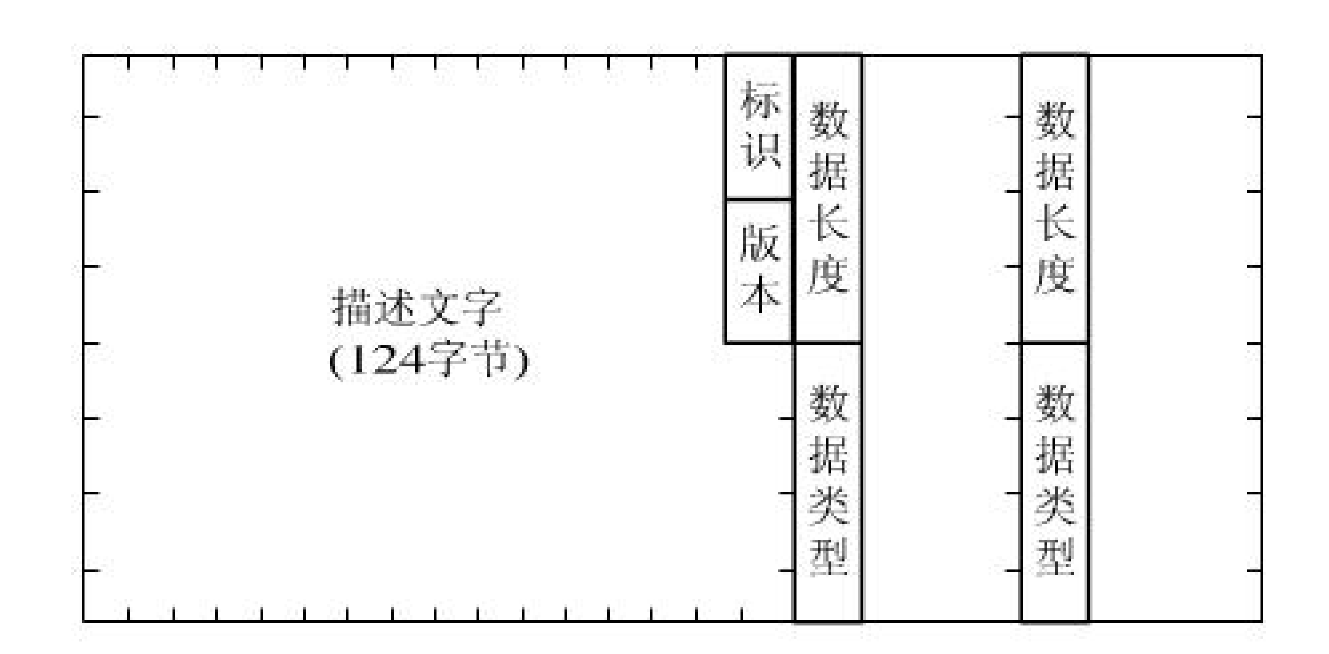


图 5-1　MAT 文件格式

相对于数据导出，数据文件的读取往往是进行文件 I/O 比较重要的环节。在 MATLAB 中进行数据文件的导入有三种不同的途径，分别是高级例程、低级例程和文件导入向导。

其中，MATLAB 的高级文件 I/O 例程分别针对不同的数据文件格式提供了不同的文件 I/O 函数，这些函数能满足大多数情况下数据导入和导出的需要。

而低级文件 I/O 例程则为访问任何一种类型文件的数据提供了接口，不过前提是编程人员必须知道文件的格式。否则读取的结果是错误的。

文件的导入向导是供数据导入用的图形界面，用于导入已知格式的文件和有一定规则的 ASCII 文本文件。

本章将分别针对不同类型的导入(导出)文件的方法进行详细的讲解。

5.2　高 级 例 程

MATLAB 的高级文件 I/O 例程提供了多种格式数据文件导入到 MATLAB 工作空间，或者将 MATLAB 工作空间保存起来的函数。针对不同的文件格式，MATLAB 提供了不同的导入、导出函数。本小节将分别讨论 MAT 格式文件、文本格式文件和一般二进制格式文件的导入、导出函数。

5.2.1　一般数据文件操作

如前文所示，MAT 文件是 MATLAB 独有的文件格式，提供了跨平台的数据交换能力，也是 MATLAB 用户最常用的数据文件格式。在 MATLAB 中，可以将当前工作空间中的变量保存成 MAT 文件，也可以将 MAT 文件中的数据导入到 MATLAB 的工作空间中，这两个过程分别使用 save 指令和 load 指令就可以完成。

需要说明一点，load 和 save 指令不仅能够加载和保存 MAT 格式的数据文件，而且还能够加载一般的文本格式文件。但是在文本数据文件中不能包含特殊的文件间隔符。因此，将这两个指令称之为操作一般数据文件的指令。

save 指令能够将当前工作空间中的变量保存到指定的数据文件中，其基本语法为

save	将当前工作空间中所有的变量保存到 matlab.mat 文件中
save filename var1 var2……	将当前工作空间中的变量 var1、var2 等保存到指定文件中
save filename data*	(功能同上)其中*为通配符

save filename　　将当前工作空间中所有的变量保存到指定的文件中

save ……option　　按照 option 的不同取值保存数据

save('filename',……)　　save 指令的函数格式用法

其中，option 可以有如下几种可能：

-append　　在已有的数据文件尾部追加数据

-ascii　　保存为 ASCII 文本格式，数据按照八位精度保存

-ascii -double　　保存为 ASCII 文本格式，数据按照十六位精度保存

-ascii -tabs　　保存为 ASCII 文本格式，数据之间使用制表符作为间隔

-ascii -doube -tabs　　上述几种选项的结合

-mat　　保存为二进制的 MAT 文件格式(默认)

-v4　　保存为版本 4 格式的数据文件

注意：

版本 4 的 MAT 文件是早期的 MATLAB 数据文件格式，现在已经很少使用了。若保存数据为文本格式时不指定间隔符为制表符，则数据之间使用空格作为数据之间的间隔。

load 指令将数据文件的数据导入到 MATLAB 的工作空间，其基本的语法为

load　　将 matlab.mat 文件中所有的变量加载到当前的工作空间

load filename　　将指定文件中所有的变量加载到当前的工作空间

load filename var1 var2 ……　　将指定文件中指定的变量加载到当前工作空间

load filename -ascii　　将数据文件按照文本格式加载

load filename –mat　　将数据文件按照 MAT 文件格式加载

S = load(……)　　load 指令的函数格式用法

注意：

使用 load 指令加载数据文件时，数据文件只要保存在 MATLAB 的搜索路径上即可，同时若不指明数据文件的扩展名，则数据文件默认按照二进制的 MAT 文件格式加载，否则都按照文本格式文件加载。

这里结合具体的操作示例来说明 save 和 load 指令的使用方法。

例子 5-1　save 和 load 指令示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> clear all
>>%创建变量
>> x1 = 2; x2 = 3; x3 = 4; y1=0;
>> %保存数据
>> save xdata x1 x2
>> %察看当前路径下的 MAT 文件
>> dir *.mat
xdata.mat
>> %将数据保存为 ASCII 格式文件
>> save xdata2.dat x* -ascii
>> clear all
```

```
>> %加载数据——默认加载二进制格式文件
>> load xdata
>> whos
  Name          Size                    Bytes  Class
  x1            1x1                         8  double array
  x2            1x1                         8  double array
Grand total is 2 elements using 16 bytes
>> %加载 ACSII 格式的数据
>> load xdata2.dat
>> whos
  Name          Size                    Bytes  Class
  x1            1x1                         8  double array
  x2            1x1                         8  double array
  xdata2        3x1                        24  double array
Grand total is 5 elements using 40 bytes
```

例子 5-1 演示了使用 save 和 load 指令保存加载数据的基本过程。需要注意，保存数据文件时的通配符“*”的使用，在例子 5-1 的操作中，保存文本格式文件时使用了该通配符，于是 MATLAB 将所有以 x 开头的变量保存了起来。另外，在加载文本格式的数据时，将所有的数据保存在一个变量中。同样，保存数据的时候，若不保存为二进制格式的 MAT 文件，则最好指定数据文件的扩展名。

注意：

Mathworks 公司推荐用户在使用 MATLAB 的过程中尽量使用 MAT 文件保存用户的数据，这样便于不同平台下的用户共享数据。

5.2.2 文本文件操作

前面小节介绍的 load 和 save 指令只能用于读写简单格式的文本文件，在很多时候，数据文件的数据之间使用了特殊的间隔符，或者数据文件直接使用 Excel 电子表格形式进行保存，这时 save 或者 load 指令就不能发挥作用了，于是，MATLAB 提供了相应的函数用来完成这些类型数据文件的读写工作。在表 5-1 中对常用数据文件的读写函数进行了总结。

表 **5-1**　常用数据文件的读写函数

文件类型	函　数	说　　明
文本文件	csvread	读取以逗号作为间隔符的文本文件
	csvwrite	保存数据到文本文件，逗号作为间隔符
	dlmread	按照指定的间隔符读取文本文件的数据
	dlmwrite	按照指定间隔符将数据写入文本文件
	textread	按照指定的格式从文本文件中读取数据
Excel 电子表格	xlsfinfo	获取文件类型等基本信息
	xlsread	读取 Excel 电子表格文件的数据
Lotus 1-2-3 电子表格	wk1read	从 Lotus1-2-3 电子表格中读取数据
	wk2write	将数据写入 Lotus1-2-3 电子表格

下面结合具体的例子来说明这些函数的用法。

例子 5-2 有间隔符的文本读写。

```
function delimiter_examp
% DELIMITER_EXAMP
% 读取具有不同间隔符号的文本数据文件
% 创建数据
A = round(rand(2,5)*100);
% 将数据 A 保存到 csvexamp.txt
csvwrite('csvexamp.txt',A);
% 在从该文件中读取数据
B = dlmread('csvexamp.txt',',');
% 进行数据处理......
% 将数据 B 保存到 dlmexamp.txt，间隔符由用户输入
c = input('输入符号作为间隔符：','s');
dlmwrite('dlmexamp.txt',B,c);
disp('保存数据文件完毕！');
%显示文件的内容
disp('csvexamp.txt:')
type csvexamp.txt
disp('dlmexamp.txt:')
type dlmexamp.txt
```

执行例子 5-2 的代码，在 MATLAB 命令行中键入：

```
>> delimiter_examp
输入符号作为间隔符：Q
保存数据文件完毕！
csvexamp.txt:

56,67,31,92,90
93,24,66,59,4

dlmexamp.txt:

56Q 67Q 31Q 92Q 90
93Q 24Q 66Q 59Q 4
```

注意：

在上面例子运行过程中，输入的间隔符 Q 后面有一个空格。

例子 5-2 中使用了 csvwrite、dlmread 和 dlmwrite 函数进行了文本文件的读写。在读写过程中，需要注意不同文件数据的间隔符号。csvread 和 csvwrite 函数可以看作是 dlmread 和 dlmwrite 函数的特殊版本。

在从文本文件中读取数据的函数中，textread 函数是一个比较特殊的函数，它能够按照用户的需要从文本文件中读取指定格式的数据。该函数能够读取的文本文件可以包含任何字符，同时，制定格式的时候可以采用 C 语言中 fscanf 使用的格式化字符串。

例子 5-3　使用 **textread** 函数。

假设在 MATLAB 的搜索路径下有一个纯文本文件包含了不同的信息，若须读取该文件的内容，可以在 MATLAB 命令行中键入下面的指令：

```
>> type season.txt
Broncos  14   2  0.8750  y
Falcons  14   2  0.8750  y
Lions     5  11  0.3125  n
Patriots 15   1  0.9375  y
Vikings   9   7  0.5625  y
>> [team, w, l, wp, playoff] = textread('season.txt', '%s %d %d %f %c')
team =
    'Broncos'
    'Falcons'
    'Lions'
    'Patriots'
    'Vikings'
w =
    14
    14
     5
    15
     9
l =
     2
⋮
>> whos
  Name        Size          Bytes  Class
  l           5x1              40  double array
  playoff     5x1              10  char array
  team        5x1             368  cell array
  w           5x1              40  double array
  wp          5x1              40  double array
Grand total is 59 elements using 498 bytes
```

例子 5-3 中使用 textread 函数从文件中读取了数据，注意数据是按照列向量读取的。一般地，使用 textread 函数读取数据时，若读取的是数字则输出为双精度的数组，否则，一般为元胞数组。所以在例子 5-3 中，读取的变量 team 是元胞数组。

textread 函数还有更复杂的用法，有兴趣的读者可以参阅 MATLAB 的帮助文档，本书就不再赘述。

例子 5-4　读取 Excel 电子表格文件的数据。

在本例中使用的电子表格文件包含下列数据：

日期	数据	这里呢？
1	11	
2	12	
3	13	
4	14	
5	15	
6	16	
7	NaN	
8	Inf	
9	19	

那么在 MATLAB 中读取该电子表格文件中的数据：

```
>> [a,b]=xlsread('xlsexamp.xls')
a =
   NaN   NaN
     1    11
     2    12
     3    13
     4    14
     5    15
     6    16
     7   NaN
     8   NaN
     9    19
b =
    '日期'    '数据'    '这里呢？'
        []        []            []
        []        []            []
        []        []            []
        []        []            []
        []        []            []
        []        []            []
        []     'NaN'            []
        []     'Inf'            []
>> whos
  Name      Size                   Bytes  Class
  a         10x2                     160  double array
  b         9x3                      416  cell array
Grand total is 61 elements using 576 bytes
```

利用 xlsread 函数从电子表格中读取数据时，一般将所有数字量读取出来放置在双精度的数组中，当单元格包含字符的时候，读取的数据为 NaN(例如读取的数据 a)。函数的第二个输出是所有单元格包含的字符串，这些字符串组成一个元胞数组，例如读取的数据 b。在读取数据的时候，需要注意 Excel 文件的版本，不是所有的 Excel 文件都能够被 MATLAB 读取的，特别是那些包含了特殊字符的文件。所以，xlsread 函数仅仅能完成一些简单的数据读取功能，比较复杂的电子表格读取可以使用 MATLAB 产品家族中的 Excel Link 工具箱。关于 Excel Link 工具将在《MATLAB 应用程序集成与发布》一书中详细介绍。

5.2.3　导入其他类型的数据文件

除了前面讲述的几种数据文件类型以外，MATLAB 还能够加载其他不同类型的数据文件，例如声音、图像等二进制数据文件。MATLAB 能够读入的二进制文件类型以及相应的加载函数信息可以通过在 MATLAB 中键入 help fileformats 命令来获取。在表 5-2 中，总结了 MATLAB 能够加载的常见的数据文件格式。

表 **5-2**　**MATLAB** 能够加载的数据文件格式

文 件 类 型	扩 展 名	说　　明
声音格式文件	.wav	Microsoft 音频格式文件
	.au	Sun 系统音频格式文件
影片格式文件	.avi	多媒体文件格式
图形图像格式	. .cur . if .　f .ico .　　. . . c . . n . n . .ras .tif .tiff . w	各种常用的图形图像格式文件
科学数据格式	.c f .　f	这里的　f 格式文件不是图像文件格式

MATLAB 为每一种格式的数据文件都准备了不同的读取函数，这些函数没有在这里列出，在 MATLAB 工作空间中键入 help fileformats 的时候会给出相应的提示。MATLAB 还提供了一个函数能够将这些函数都导入到 MATLAB 工作空间，这个函数就是 importdata，首先来察看一个例子。

例子 5-5　importdata 函数的使用示例。

在当前的目录中有三个数据文件，其中一个是声音文件为 train.wav，一个是图像文件为 sample.jpg，另外一个为例子 5-4 中使用的 Excel 电子表格，这里统一使用 importdata 函数将它们导入。在 MATLAB 命令行中键入下面的指令：

```
c ar a
```

```
>> %导入声音文件
>> snd = importdata('train.wav');
>> whos
  Name        Size                     Bytes    Class
  snd         1x1                      103296   struct array
Grand total is 12883 elements using 103296 bytes
>> %将声音播放出来
>> sound(snd.data,snd.fs)
>> %导入图像文件
>> img = importdata('sample.jpg');
>> whos
  Name      Size                       Bytes    Class
  img       473x600x3                  851400   uint8 array
  snd       1x1                        103296   struct array
Grand total is 864283 elements using 954696 bytes
>> %在图形窗体中显示图像
>> image(img)
>> %导入 Excel 电子表格
>> xls = importdata('xlsexamp.xls');
>> whos
  Name      Size                       Bytes    Class
  img       473x600x3                  851400   uint8 array
  snd       1x1                        103296   struct array
  xls       1x1                           824   struct array
Grand total is 864346 elements using 955520 bytes
>> %xls 的内容
>> xls
xls =
         data: [10x2 double]
     textdata: {9x3 cell}
```

导入的图片文件在 MATLAB 图形窗体中的显示效果如图 5-2 所示。

例子 5-5 使用 importdata 函数导入了三种不同格式的数据文件，并且利用不同的方式将相应的数据显示出来，比如声音文件通过声卡播放出来，而图像文件则通过图形窗体显示出来。importdata 函数可以看作是导入数据的万能函数，该函数几乎可以导入 MATLAB 支持的各种格式类型的数据文件。该函数通过函数 finfo 获取数据文件的类型信息，然后使用不同的辅助函数来加载不同的数据文件，例如加载图像文件使用 imread 函数，加载声音文件则使用 auread 函数等。使用这个函数的好处非常明显，就是利用一个函数就可以完成加载各种数据的操作，不过也有相应的缺点，就是程序的效率不比使用专门的函数好。有兴趣的读者可以尝试读读 importdata 函数的源代码，这里给出部分代码片段。

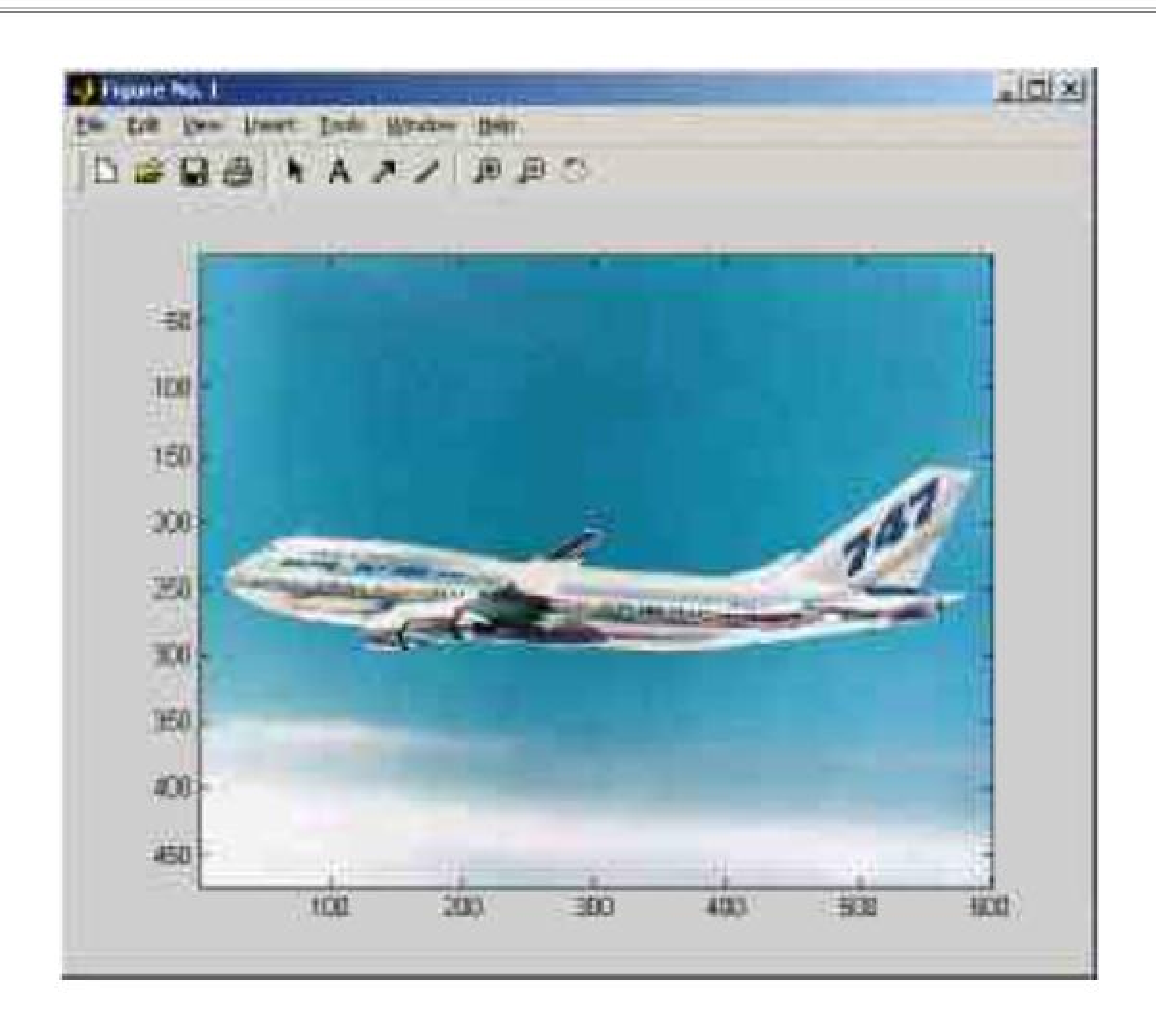


图 5-2　加载的图片文件的显示效果

例子 5-6　importdata 函数源代码清单片段。

```
⋮
% 读入 Lotus 电子表格文件
case 'wk1'
 [out.data, out.textdata] = wk1read(FileName);
 out = LocalRowColShuffle(out);
% 读入 AVI 影片格式文件
case 'avi'
 out = aviread(FileName);
% 读入图形图像格式文件
case 'im'
 [out.cdata, out.colormap] = imread(FileName);
% 读入 AU 格式声音格式文件
case {'au','snd'}
 [out.data, out.fs] = auread(FileName);
% 读入 WAV 格式声音文件
case 'wav'
 [out.data, out.fs] = wavread(FileName);
⋮
```

关于表 5-3 中函数的具体用法可以参阅 importdata 函数的源代码和帮助文档，这里就不作详细解释了。

表 **5-3**　部分数据文件的专用加载函数

文件类型	扩 展 名	函 数	输出数据格式
特殊科学数据格式	CDF	cdfread	元胞数组
	FITS	fitsread	主/副数据表集合
	HDF	hdfread	HDF/HDF-EOS 数据
图形图像格式	BMP、JPG、TIFF 等	imread	色彩数据和灰度/色彩索引数组
声音格式文件	WAV	wavread	声音数据和采样率
	AU	auread	
影片格式文件	AVI	aviread	MATLAB 影片格式文件

5.2.4　导出二进制格式数据

MATLAB 除了能够导入前面小节介绍的各种格式的数据文件外，还可以将工作空间的数据导出成不同格式的文件。这一过程不仅可以通过图形用户界面完成，还可以通过 MATLAB 函数来完成。不过，导出二进制文件没有统一的函数可以使用，不同格式的文件有不同的导出函数，在表 5-4 中对部分函数进行了总结。

表 **5-4**　二进制数据文件导出函数

文件类型	扩展名	函数
声音文件	AU	auwrite
	WAV	wavwirte
图像文件	BMP、JPG 等	imwrite
影片格式文件	AVI	avifile
特殊科学数据格式	CDF	cdfwrite
	HDF	使用图形用户界面导出

注意：

这里使用高级文件 I/O 函数导出数据文件，对于 MATLAB 目前不支持的数据文件格式可以使用 5.3 小节介绍的低级文件 I/O 例程导出数据，但是前提是必须知道相应的文件格式。另外，若扩展名为 .hdf 的文件为图形文件时，则使用 imwrite 函数导出。

在本小节，使用一个将图片文件导出成为 AVI 格式文件的例子来说明导出二进制文件的过程。

例子 5-7　导出数据为 AVI 文件。

```
function avi e am
  AVI E  AMP  导出数据为 AVI 格式

   创建 AVI 文件对象
avio     avifile m movie avi  f  s
   为 AVI 文件添加帧数据
for    1
```

```
h = plot(fft(eye(k+16)));
set(h,'EraseMode','xor');
axis equal;
% 获取当前帧数据
frame = getframe(gca);
% 添加帧数据到 AVI 文件
aviobj = addframe(aviobj,frame);
end
% 关闭 AVI 文件句柄
aviobj = close(aviobj);
```

在 MATLAB 命令行窗口中运行该函数，运行过程中将连续绘制 25 个不同的图像，运行结束后，在当前的路径下将创建一个 AVI 文件——mymovie.avi。可以利用 Windows Media Player 将文件播放出来，如图 5-3 所示。

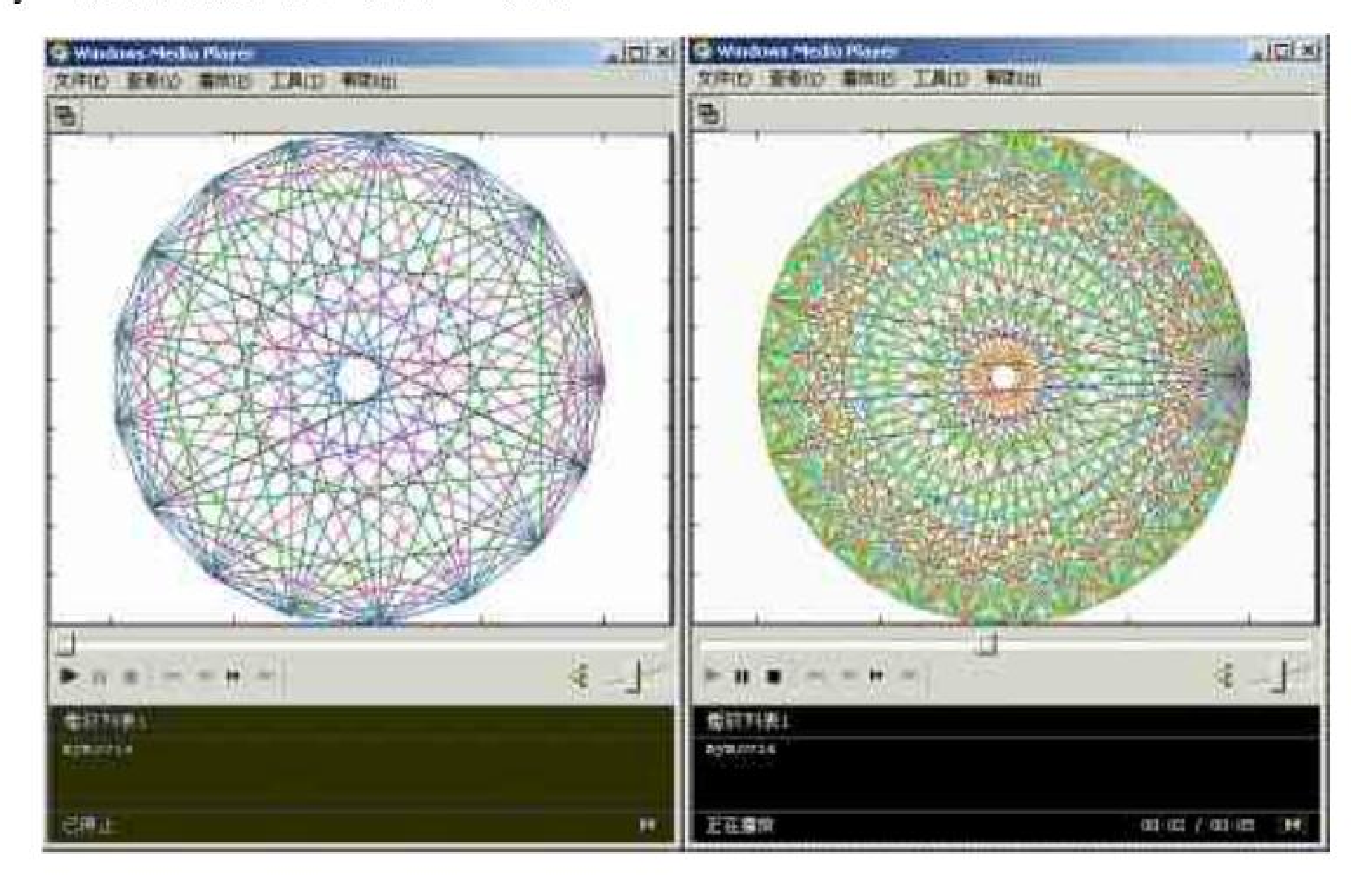


图 5-3　播放 AVI 文件

将 MATLAB 的数据导出为 AVI 文件的过程主要有三个步骤：

(1) 用 avifile 函数创建 AVI 文件的对象，获取文件的句柄。

(2) 通过 addframe 函数将必要的图形文件按照每一帧添加到 AVI 文件中。

(3) 用 close 函数关闭 AVI 文件句柄。

例子 5-7 演示了创建 AVI 文件的全过程。

关于其他二进制文件的导出过程本章就不再讲述了，有兴趣的读者可以参阅 MATLAB 的帮助文档。

5.3　低 级 例 程

MATLAB 提供了一组函数用于低级文件的 I/O 操作。所谓低级例程，是指在使用这些函数进行文件读写的时候需要程序员了解文件的格式，并且按照文件的格式进行相应的数据转换。MATLAB 的低级文件 I/O 例程是以 ANSI C 语言标准库函数中文件 I/O 函数为基础开发的，所以，从形式上看，这些函数和 C 语言文件的 I/O 函数没有明显的区别，如果读者对 C 语言的 I/O 函数比较了解，那么在理解 MATLAB 低级文件 I/O 函数上没有任何的障碍。

利用低级例程进行文件的 I/O 主要有三个步骤：

(1) 打开文件，把文件的 ID 保存到一个变量中。

(2) 对文件的数据进行读或者写。

(3) 关闭文件。

注意：

一旦将文件打开，即使在文件读写的过程中发生了错误，文件也会一直保持打开的状态，所以，当再次打开没有关闭的文件并从中读取或者写入数据时，就会出现严重的错误——最严重的情况会导致内存分配故障、泄漏和内存崩溃。

若在脚本 M 文件中进行文件 I/O 时，只要能找到文件的 ID 就能够处理此类错误。但是，若在函数中就比较麻烦了，因为文件的 ID 号并不存在于基本工作区之中。为了避免遇到这样的问题，可以将 Editor/Debugger 设置为 Stop If Error，这时若出现了错误就可以通过访问函数的工作区(工作空间中包含文件的 ID)在继续工作前关闭该文件。

5.3.1　打开关闭文件

在进行文件读写之前必须将需要读写的文件在 MATLAB 中打开。打开文件的函数为 fopen，其命令行格式为

```
fid = fopen('filename', 'flag');
```

其中，flag 为控制文件读写的标识符，它的取值可以为

- r 表示打开的文件进行读的操作。
- w 表示打开的文件进行写的操作，若文件不存在则创建新的文件。
- a 表示打开的文件进行追加数据的操作，若文件不存在则创建新的文件。
- r+表示打开的文件既可以进行写的操作，也可以进行读的操作。
- w+表示打开的文件既可以进行写的操作，也可以进行读的操作，若文件不存在则创建新的文件。
- a+表示打开的文件既可以进行写的操作、读的操作，也可以进行数据追加操作，若文件不存在则创建新的文件。

注意：

在 Windows 平台中打开文件的时候需要进一步指定文件类型——二进制文件或者文本文件，例如打开一个只读的文本文件时，flag 应该写作 rt，而打开可读写的二进制文件时，flag 应该为 rb+。

若不指定 flag，则 MATLAB 按照只读形式打开二进制类型文件。

在打开只读文件时，filename 指定的文件只要存在于 MATLAB 的搜索路径中即可。

若能够成功地打开文件，则 fid 为非负的整数，否则为- 1，而相应的错误信息，可以作为 fopen 函数的第二个输出参数输出到工作空间中。使用 fopen 函数的常见代码段如下所示：

```
fid=0;
filename=input('Open file: ', 's');
[fid,message] = f pen(filename, ' ');
if fid ==   1
    disp(message)
end
```

这段代码从命令行窗口中获取一个文件名，然后用 fopen 函数打开它，若没有成功，则将相应的错误信息显示在命令行窗口中。

关于 fopen 函数的详细用法请参阅 MATLAB 的帮助文档。

关闭已经打开的文件需要使用 fclose 函数，其基本的命令格式为

status = fclose(fid)

关闭文件之后，fid 变量依然存在于工作空间中，但是对 fid 再进行文件 I/O 操作是错误的。若函数运行成功，则 status 为 0，否则为- 1。

5.3.2 读写数据

打开文件之后就要进行文件内容的读写了，MATLAB 提供了两大类低级文件 I/O 函数进行文件内容的读写——二进制文件读写函数和文本文件读写函数，在表 5-5 中对这些函数进行了简要的总结。

表 **5-5**　读写数据的低级 **I/O** 函数

函　数	说　　明	输　　出
fs anf	从文件中读取格式化的输入	数据矩阵
fp intf	向文件写入格式化的输出	写入数据文件的数据个数
fgetl	读取文本文件的一行数据，不包含文本的结束符	字符串
fgets	读取文本文件的一行数据，包含文本的结束符	字符串
f ead	读取文件的二进制数据	数据矩阵和读入的数据个数
f ite	写入文件的二进制数据	写入的字节数

若在文件读写过程中出现了错误，可以使用 ferror 函数获取文件 I/O 过程的错误信息。下面通过针对不同类型的文件的读写举例来说明这些低级文件 I/O 函数的使用方法。

例子 5-8　格式化输入、输出示例。

```
fun ti n data = t ti   e amp
         O           文本文件的格式化输入输出

  打开一个文本文件写入数据
[fid msg] = f pen('s  ua e  mat t  t','  t');
```

```
006     if fid == - 1
007         disp(msg);
008         return;
009     end
010     % 写入数据
011     fprintf(fid,'%s\n','文本文件格式化输入输出示例');
012     fprintf(fid, '%i\t%i\t%i\n',[1 2 3;4 5 6;7 8 9]);
013     % 关闭文件
014     fclose(fid);
015     % 打开文本文件读入数据
016     fid=fopen('square_mat.txt', 'rt');
017     if fid == - 1
018         disp(msg);
019         return;
020     end
021     % 读取数据
022     title = fgetl(fid);
023     disp(title);
024     data=fscanf(fid, '%i');
025     data = reshape(data, 3, 3);
026     % 关闭文件
027     fclose(fid);
```

在例子 5-8 展示了使用 fscanf 和 fprintf 进行数据文件 I/O 的过程，这两个函数使用起来和 C 语言的函数没有太多区别，其格式化的文本和 C 语言的也保持一致，具体的请参阅 C 语言的说明或者 MATLAB 的帮助文档。

不过在 MATLAB 中使用这些函数充分利用了基于向量或者矩阵的运算特点，例如在例子 5-8 的 012、024 行进行数据的写入和读取操作时，若使用 C 语言完成同样的工作则需要使用循环来处理，但是在 MATLAB 中仅仅用一行代码就实现了相同的工作。运行例子 5-8，在 MATLAB 命令行中键入如下指令：

```
>> [data count] = txtio_examp
文本文件格式化输入输出示例
data =
     1     2     3
     4     5     6
     7     8     9
count =
    18
```

可以看出，在这里读取的整数每个元素占用了两个字节的内存空间。

例子 5-9　二进制文件的读写。

```
001 function [data,count,status] = binio_examp
002 %BINIO_EXAMP 二进制文件读写示例
003
004 % 打开二进制文件写入数据
005 fid = fopen('magic5.bin','wb');
006 % 写入文本数据
007 count = fwrite(fid,'喂，你好吗？','int32');
008 % 写入数据
009 fwrite(fid,magic(5),'int32');
010 %关闭文件
011 status = fclose(fid);
012 % 打开二进制文件读取数据
013 fid = fopen('magic5.bin','rb');
014 % 读取文本
015 S = fread(fid,count,'int32');
016 disp(['读取数据类型: ',class(S)]);
017 disp(['读取数据内容: ',char(S')]);
018 % 读取数据
019 [data count] = fread(fid,'int32');
020 data = reshape(data,5,5);
021 %关闭文件
022 status = fclose(fid);
```

读写二进制文件的时候略微麻烦一些，就是在读写数据的时候需要指定数据的类型和读取数据的个数。例如在例子 5-9 的 007、009 行写入数据时，分别要指定写入数据的类型，在 015、019 行读入数据时，需要指定读取的数据类型和个数，并且这些信息要同数据文件内容保持一致，否则读入的数据就不会正确。

表 5-6 中总结了二进制文件读写时的数据类型标识符。

表 **5-6**　数据类型标识符

标识符	说　明	标识符	说　明
schar	有符号的字符，8 位数据	uint16	16 位无符号整数
uchar	无符号的字符，8 位数据	uint32	32 位无符号整数
int8	8 位有符号整数	uint64	64 位无符号整数
int16	16 位有符号整数	float32	32 位浮点数
int32	32 位有符号整数	float64	64 位浮点数
int64	64 位有符号整数	double	双精度数 64 位数据
uint8	8 位无符号整数		

由于在写入二进制文件数据时只能够写入 8 位的字符数据，所以在例子 5-9 写入文本数据的时候没有将文本数据按照字符类型写入，而是按照 32 位整数的格式写入的，也可以按照 16 位无符号整数类型写入，所以在读入数据时也按照 32 位整数的格式读入。若在 fread

函数或者 fwrite 函数进行操作的时候不指定数据类型标识符，则默认按照 uchar 的格式读写数据。运行例子 5-9 的代码，在 MATLAB 命令行中键入：

```
>> [data count status] = binio_examp
读取数据类型: double
读取数据内容: 喂，你好吗?
data =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
count =
    25
status =
     0
```

除了在表 5-6 中总结的各种数据类型标识符外，还可以使用 C++语言的数据类型关键字来表示不同的数据类型，例如 short、float、ushort、long 等，只不过在使用这些关键字的时候，不同的操作系统平台上可能同样的数据类型占用的字节数不尽相同，所以，尽量不使用这些数据类型关键字进行二进制数据文件的 I/O，具体的内容介绍可以参阅 C++语言手册或者 MATLAB 的帮助文档。

5.3.3　文件位置指针

当正确地打开文件并进行数据的读写时，MATLAB 自动创建一个文件位置指针来管理和维护文件读写数据的起始位置。所以，在进行数据文件的读写时，需要通过某种手段来控制和判断当前文件位置指针，例如判断当前文件位置指针是否已经到达文件尾部，将文件位置指针移动到指定的位置，获取当前文件位置指针在文件中的位置以及将文件位置指针重置在文件的头部等。在 MATLAB 中，通过表 5-7 中的函数来控制、判断文件位置指针。

表 **5-7**　文件位置指针函数

函 数	说　明
fseek	设置文件位置指针到指定的位置
ftell	获取当前文件位置指针的位置
feof	判断当前的文件位置指针是否到达文件尾部
frewind	将文件位置指针返回到文件的起始位置

其中，fseek 函数的命令行格式如下：

```
status = fseek(fid,offset,origin)
```

在命令行中，fid 指已经打开的数据文件，而 offset 是指移动文件指针的偏移量，若数值为正则向文件尾部的方向移动文件位置指针，若数值为 0 则不移动文件位置指针，若数值为负则向文件头部的方面移动文件位置指针，offset 的单位为字节数。origin 为字符串，代表文件指针的位置，有效值为 bof，表示文件的头部，cof 表示当前的文件指针位置，eof

表示文件的尾部。函数的返回值 status 若为 0 则表示操作成功，否则为- 1。错误的类型可以用 ferror 函数获取。

例子 5-10 文件位置指针函数示例。

```
function [pos,status] = pos_examp
%POS_EXAMP 文件位置指针示例

% 创建文件
fid = fopen('testdata.dat','wb');
x = 1:10;
fwrite(fid,x,'short');
fclose(fid);
% 打开数据文件
fid = fopen('testdata.dat','rb');
%获取当前的文件指针位置
pos = ftell(fid);
disp(['当前的文件位置指针:',num2str(pos)]);
% 向文件尾部移动文件指针 6 个字节
status = fseek(fid,6,'bof');
% 读取数据
four =fread(fid,1,'short');
disp(['读取的数据: ',num2str(four)]);
% 获取当前的文件指针
pos = ftell(fid);
disp(['当前的文件位置指针:',num2str(pos)]);
% 从当前的位置向文件头部移动指针 4 个字节
status = fseek(fid, - 4,'cof');
% 获取当前的文件指针
pos = ftell(fid);
disp([ '当前的文件位置指针: ',num2str(pos)]);
% 读取数据
three = fread(fid,1,'short');
disp(['读取的数据: ',num2str(three)]);
```

例子 5-10 说明了数据文件位置指针移动和获取的各种方法。运行例子 5-10，在 MATLAB 命令行窗口中键入下面的指令：

```
>> [pos,status] = pos_examp
当前的文件位置指针:0
读取的数据: 4
当前的文件位置指针: 8
```

```
当前的文件位置指针: 4
读取的数据: 3
pos =
     4
status =
     0
```

充分利用文件位置指针的功能能够更加方便地读写数据文件，请读者结合前文的文件读写函数理解例子 5-10 的代码。

5.4　文件导入向导

MATLAB 为了便于用户导入数据还提供了导入数据向导，导入数据向导是一个图形用户界面，能够帮助用户导入各种类型的数据，这样，就不必通过编写程序就可以把数据导入到 MATLAB 的工作空间。本小节通过具体的示例来讲解导入数据向导的使用方法。

例子 5-11　通过数据导入向导导入文本数据文件。

首先启动导入数据向导，启动导入数据向导有不同的方法：

- 通过菜单命令，执行“File”菜单下的“Import Data”命令。
- 在 MATLAB 命令行中，键入指令 uiimport。
- 执行 MATLAB 的“Start”菜单中“MATLAB”子菜单下的“Import Wizard”命令。

启动的导入数据向导如图 5-4 所示。

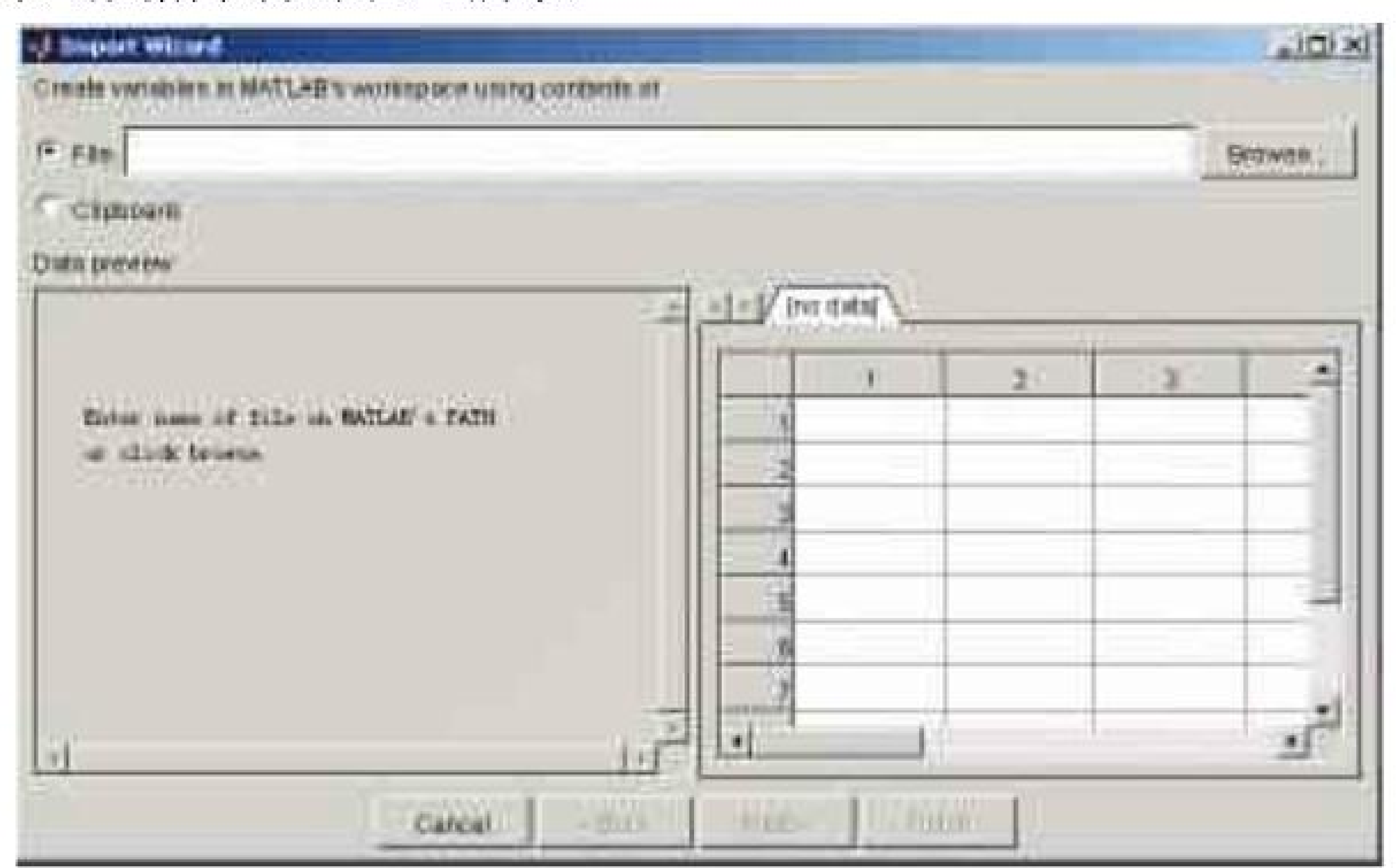


图 5-4　启动数据导入向导的初始画面

单击“Browse”按钮，在弹出的对话框中选择需要加载的数据文件，单击“打开”按钮，对话框如图 5-5 所示。

用鼠标选择需要打开的文件，然后单击“打开”按钮之后，数据导入向导尝试分析并加载数据文件，并且将能够加载的数据显示在图形界面中。



图 5-5　选择需要加载的数据文件

在如图 5-6 的对话框中可以通过“data”、“textdata”和“colheaders”标签页察看导入的数据，若导入正确无误后，单击“Next”按钮，对话框将要求用户选择数据的分隔符号，如图 5-7 所示。

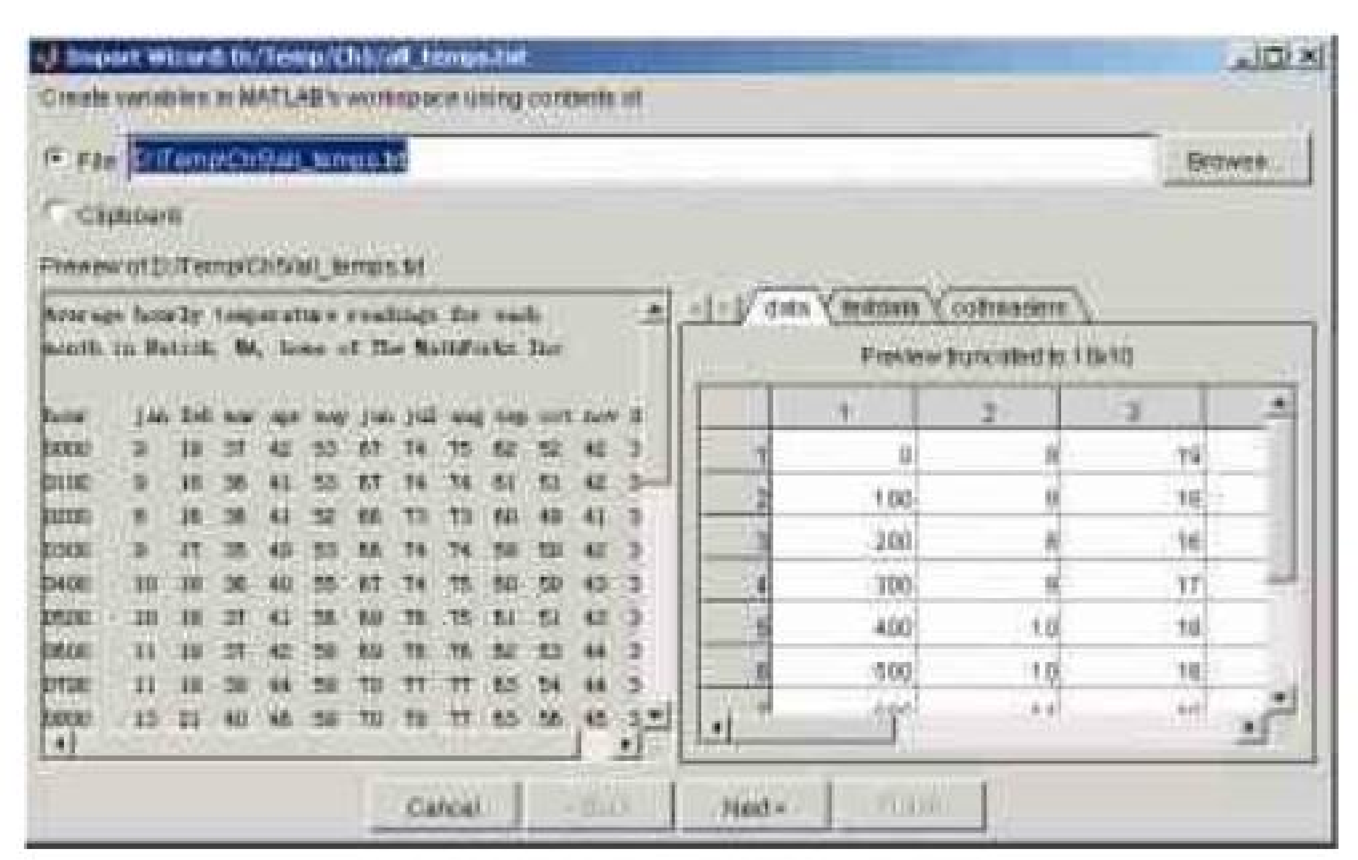


图 5-6　加载了数据的数据导入对话框

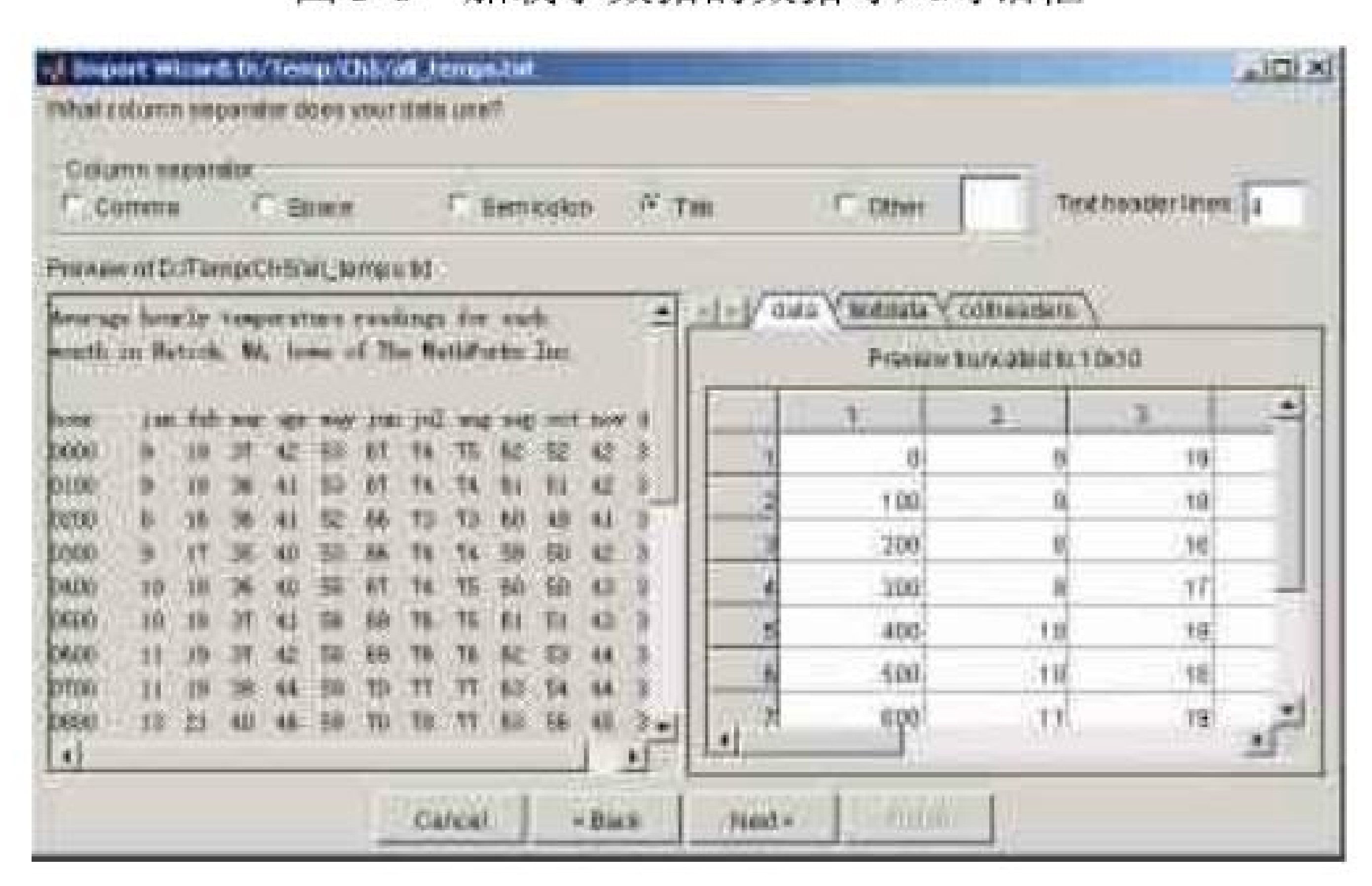


图 5-7　选择不同的分隔符号

在对话框中可以直接设置的分隔符有逗号(Comma)、空格符(Space)、分号(Semicolon)、制表符(Tab)或者自定义(Other)，当选择自定义(Other)方式时，需要在相应的文本输入框中指定分隔符字符。

不同的数据文件包含的数据分隔符不尽相同，所以不同的分隔符号在同样的数据文件

中导入的数据也不一样。在这一步骤中一定要选择合适的数据分隔符，之后单击“Next”按钮，对话框要求选择不同的数据变量，如图 5-8 所示。

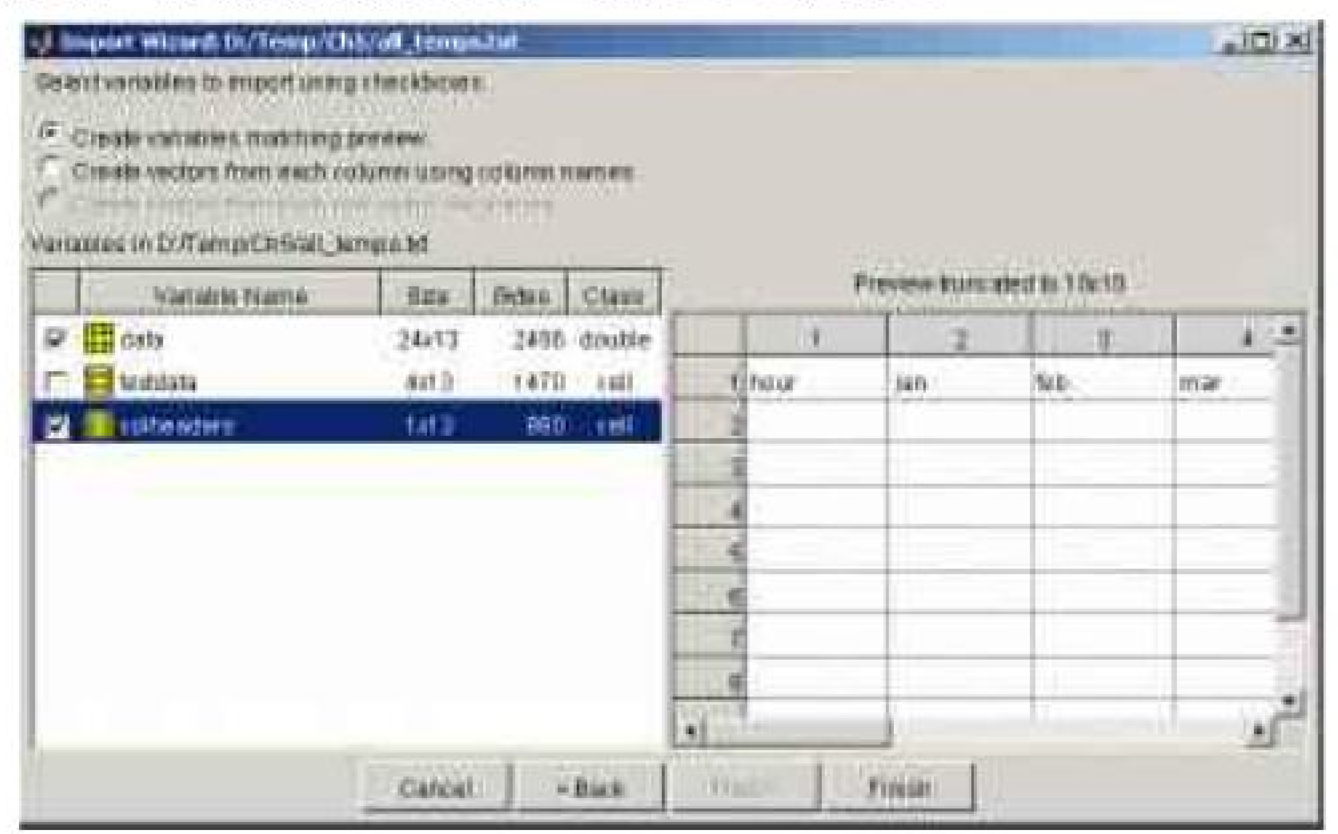


图 5-8　选择数据变量

选择数据变量之后就可以单击“Finish”按钮完成整个数据导入的过程，这时在 MATLAB 的工作空间中，将出现刚才导入的两个变量，在 MATLAB 中键入 whos 命令察看：

```
>> whos
  Name              Size                    Bytes  Class
  colheaders        1x13                      860  cell array
  data              24x13                    2496  double array
Grand total is 365 elements using 3356 bytes
```

导入数据向导不仅可以从文本文件或者二进制文件中导入数据，而且还能够从剪贴板中导入数据，例子 5-12 演示了这一过程。

例子 5-12　从剪贴板导入数据。

本例子使用的数据文件为例子 5-4 使用的 Excel 文件。首先在 Excel 中打开该文件，并且选择数据文件的 A、B、C 三列，通过“编辑”菜单下的“拷贝”命令，或者通过快捷键 Ctrl+C 拷贝数据，如图 5-9 所示。

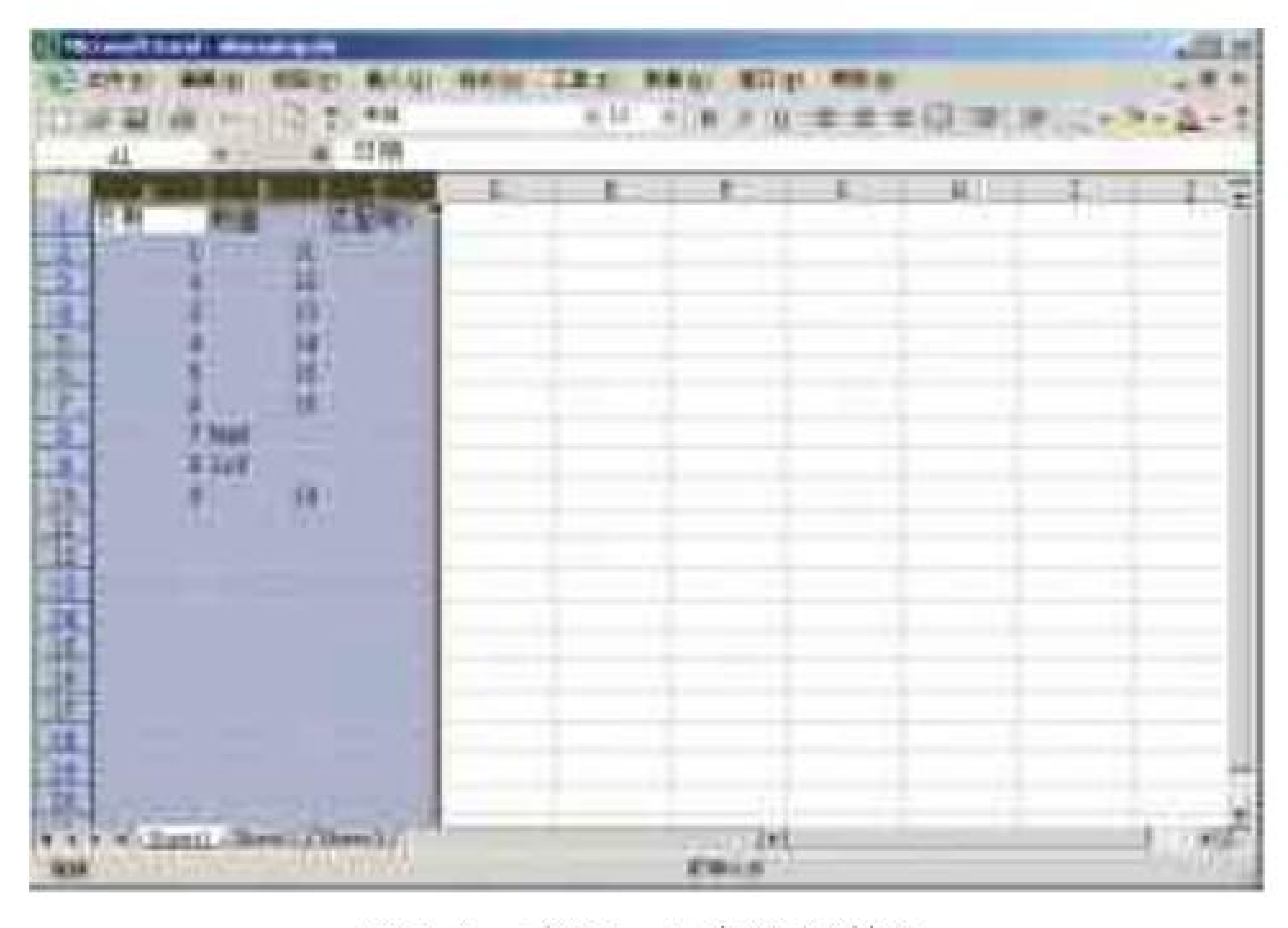


图 5-9　在 Excel 中选择数据

启动数据导入向导，在数据导入向导的第一个界面选择“Clipboard”单选框，若剪贴板上存在有效的能够导入的数据时，则数据导入向导如图 5-10 所示。

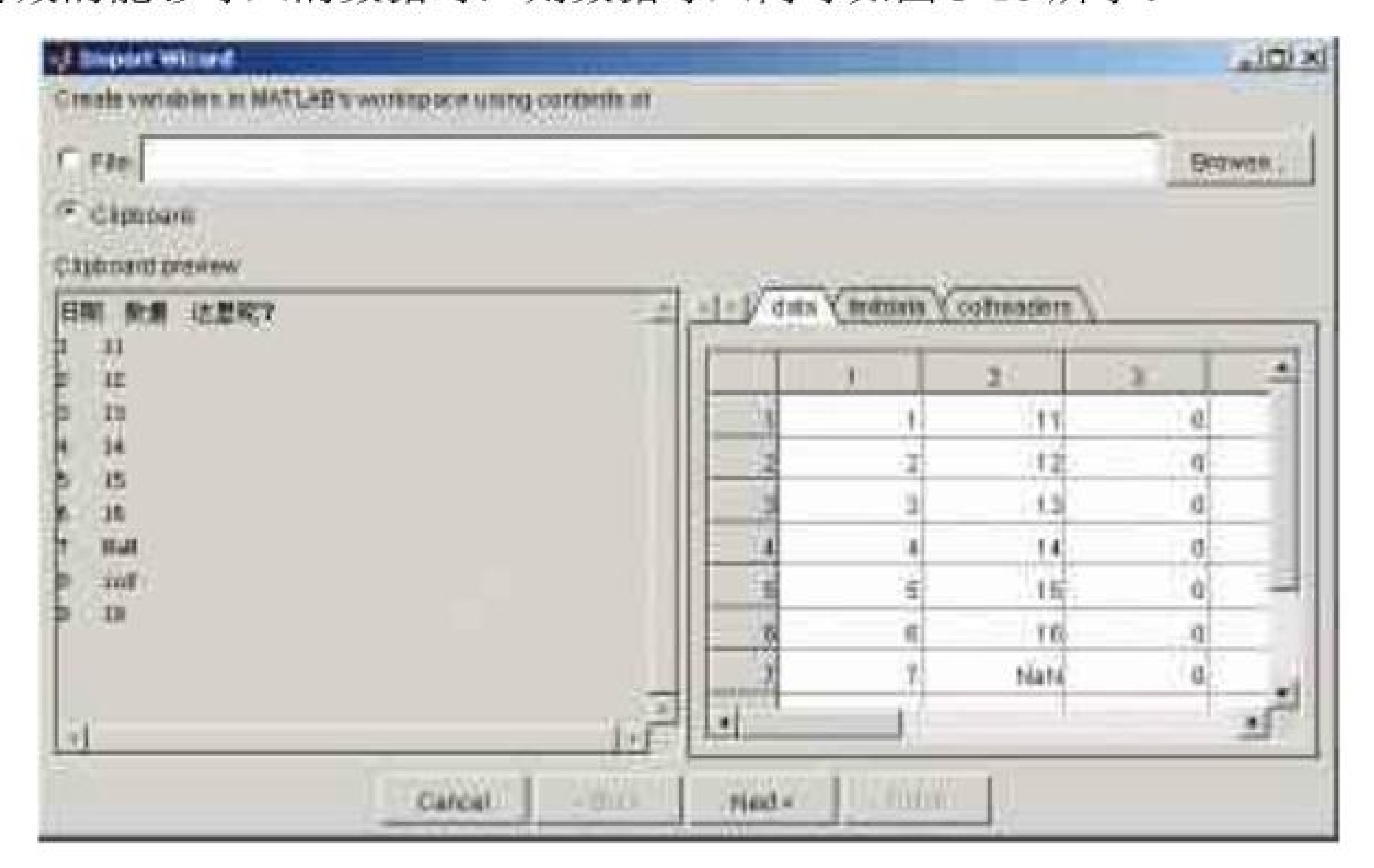


图 5-10　通过剪贴板导入数据

接下来的步骤和例子 5-11 中的一致，这里不再赘述，不过通过剪贴板导入的数据和例子 5-4 导入的数据得到的变量不同，通过例子 5-4 导入的数据变量 a 和 b，分别将 NaN 和 Inf 作为字符串处理，而利用数据导入向导导入数据的时候，将 NaN 和 Inf 分别作为数据处理。导入数据后在 MATLAB 的工作空间中察看数据：

```
>> whos
  Name        Size                    Bytes  Class
  data        9x3                       216  double array
Grand total is 27 elements using 216 bytes
>> data
data =
     1     11     0
     2     12     0
     3     13     0
     4     14     0
     5     15     0
     6     16     0
     7     NaN    0
     8     Inf    0
     9     19     0
```

注意比较变量 data 和例子 5-4 中变量 a 的差别。

例子 5-13　通过数据导入向导导入二进制文件。

数据导入向导还能够导入二进制文件，这里以声音文件作为例子演示导入数据的过程，导入其他类型的数据文件的过程与之只存在某些细节上的不同，但是总的步骤是一致的。

首先启动数据导入向导，并且选择需要导入的声音文件 train.wav，这时的导入数据向导如图 5-11 所示。

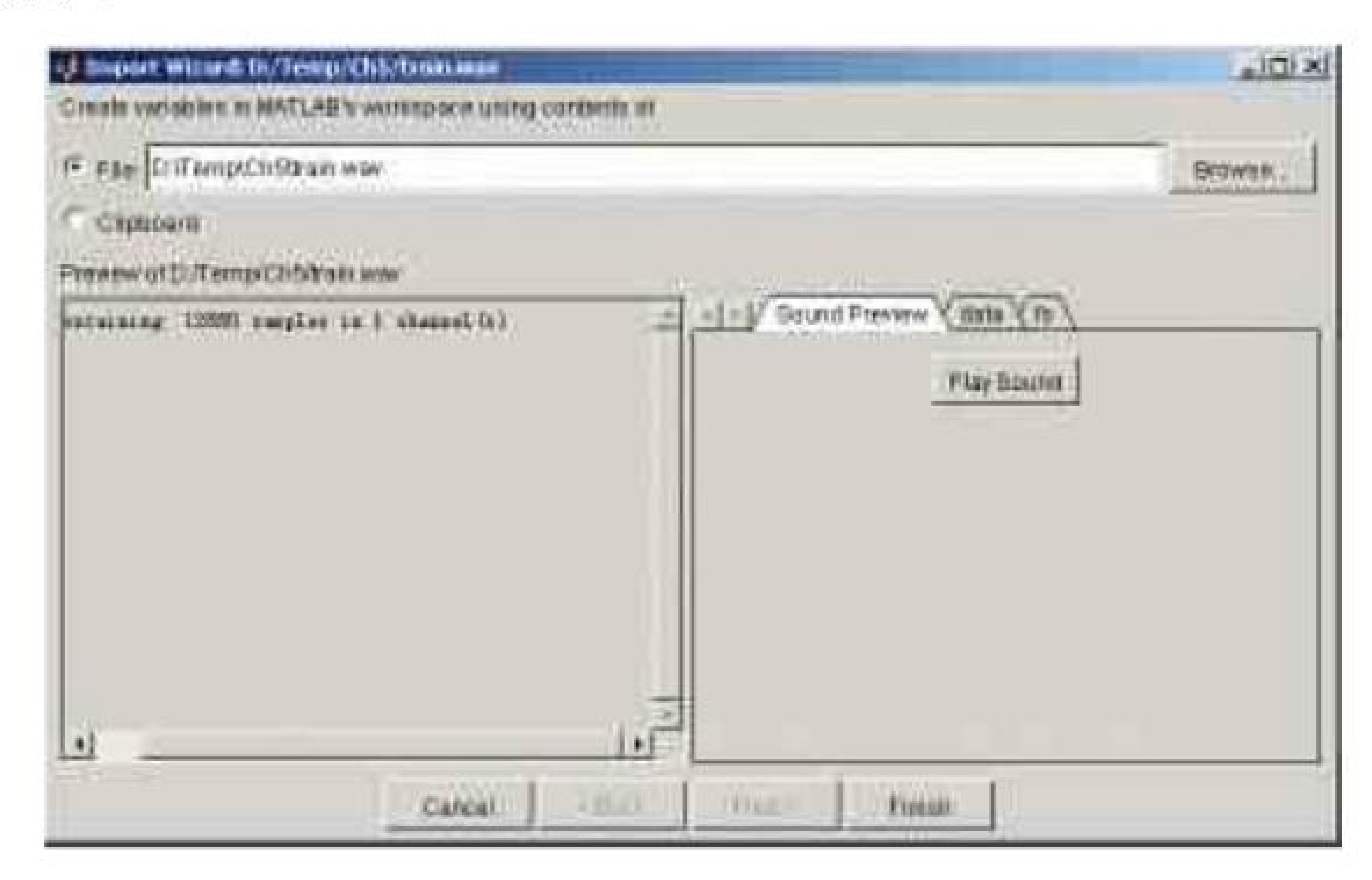


图 5-11　导入声音数据的导入数据向导

单击界面上的“Play Sound”按钮，可以通过计算机的声卡播放声音，并且能够给出声音数据的图形显示，如图 5-12 所示。

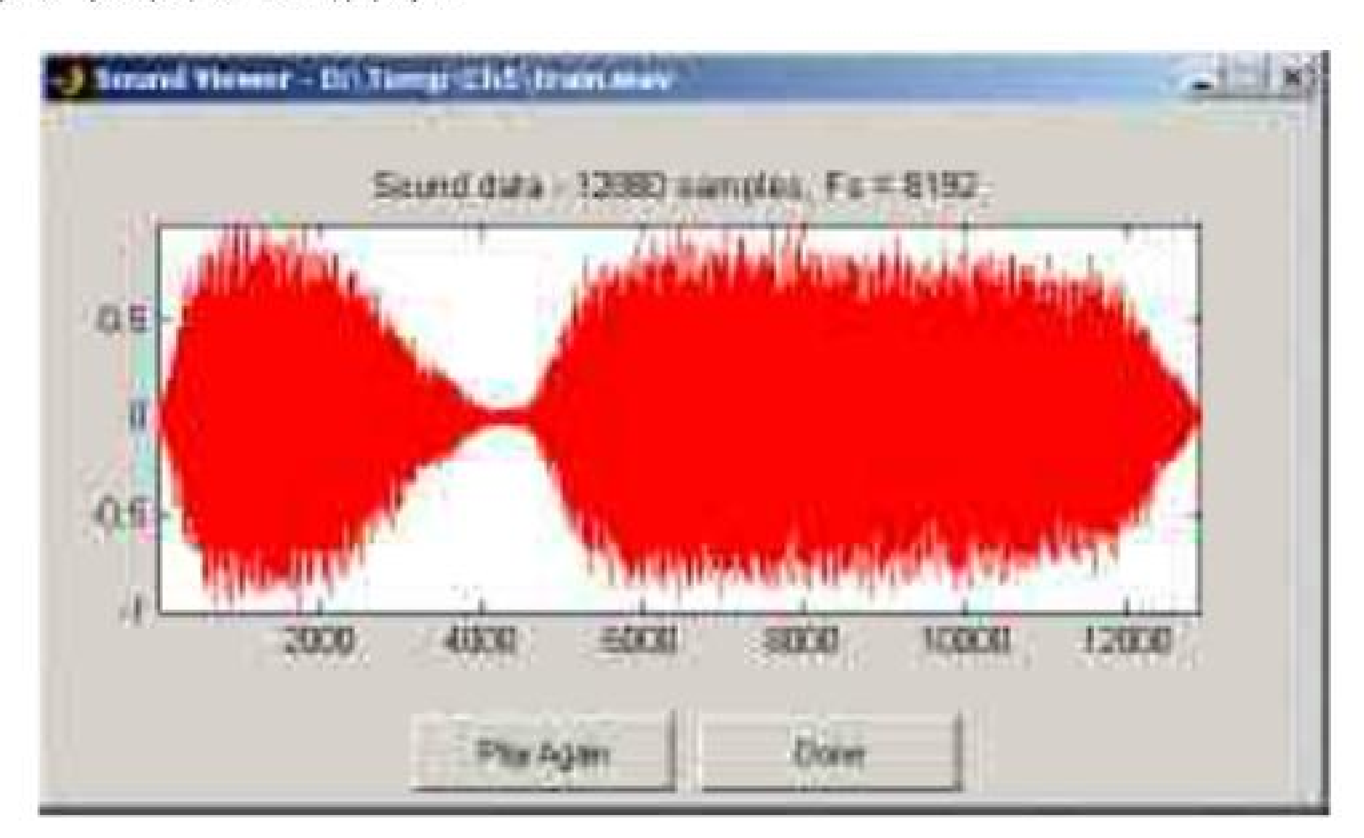


图 5-12　声音文件的图形显示

这里不仅通过图形方式显示了数据文件，还给出了声音文件的统计信息，例如声音文件的采样率和声音文件的采样点数。

在导入数据向导的“data”和“fs”标签页可以察看相应的数据，然后单击“Finish”按钮完成导入数据的过程，这时导入工作空间的数据为一个结构，通过 whos 指令察看：

```
>> whos
  Name          Size                    Bytes  Class
  train         1x1                    103296   struct array
Grand total is 12883 elements using 103296 bytes
>> train
train =
```

```
data: [12880x1 double]
  fs: 8192
```

关于利用数据导入向导导入其他类型数据的方法这里就不再赘述了，有兴趣的读者可以尝试导入不同类型的数据文件，来熟悉数据导入向导的使用，或者阅读 MATLAB 的帮助文档以了解更详细的数据 I/O 信息。

5.5　本 章 小 结

通过本章的学习，读者应该能够了解并掌握使用 MAT 文件，以及在 MATLAB 中导入各种数据文件数据的方法，还可以通过编写 M 语言代码完成数据文件的读写过程。MATLAB 的数据导入、导出功能非常强大，通过导入、导出数据，使 MATLAB 能够灵活处理各种格式的数据文件，利用 MATLAB 进行数据处理是一件非常容易的事情，但是导入数据是这一过程的前提。本章讲述了大量的 MATLAB 内建函数和 M 核心函数，读者应该通过阅读代码和自己动手实践尽快掌握这些函数的用法。

在本章，重点介绍的是 MATLAB 和外界各种数据文件进行交互的方法，其中包括各种高级例程和低级例程，最后还介绍了数据导入向导的使用方法。MATLAB 能够处理的信息都必须通过导入数据文件来完成，而 MATLAB 产品体系所包含的各种产品模块能够处理的信息远不仅仅是数据文件，例如，利用 Excel Link 工具箱可以从 Excel 软件的电子表格中获取数据，利用 Database 工具箱能够从各种关系型数据库中导入数据，利用 Data Acquisition 工具箱可以从各种类型的 PC 数据采集器中获取实测数据，利用 Instrument Control 工具箱能够从 GBIP、VXI、RS232/422 等总线中读取数据，利用 Image Acquisition 工具箱能够从图像获取设备中采集图像数据。这些工具箱极大丰富了 MATLAB 的功能，使其能够获取各种数据信息，从而利用 MATLAB 强大的数据分析、处理、建模和仿真能力构造集成化的测量、测试环境。

有关上述工具箱以及 MATLAB 在测量、测试系统方面的解决方案，请读者参阅 MATLAB 的产品介绍，或者在互联网上查阅有关的产品信息。

第六章　图 形 基 础

除去以矩阵为基本运算单位这一点外，强大灵活的数据可视化功能也是 MATLAB 最吸引人的特点之一。相比杂乱无章的数据，图形显得更加直观，便于工程师从整体上把握全局，同时也能够掌握细节。前面章节在部分内容中已经略微使用了 MATLAB 的绘图能力，在本章，将详细介绍 MATLAB 数据可视化和绘图方面的能力。

本章讲述的主要内容如下：

- 基本二维和三维绘图；
- 数据插值和曲线拟合；
- 专用绘图程序；
- 图像和[illegible]。

6.1　概　　述

数据的可视化是 MATLAB 的强大功能之一，而这仅仅是 MATLAB 图形功能的一部分，MATLAB 的图形功能主要包括数据可视化、创建用户图形界面和简单数据统计处理等，其中，数据的可视化不仅仅是二维的，还可以在三维空间展示数据，而数据或者图形的可视化也是进行数据处理或者图形图像处理的第一步骤。

MATLAB 的图形都是绘制在 MATLAB 的图形窗体中的，而所有图形数据可视化的工作也都以图形窗体为主。MATLAB 图形窗体如图 6-1 所示。

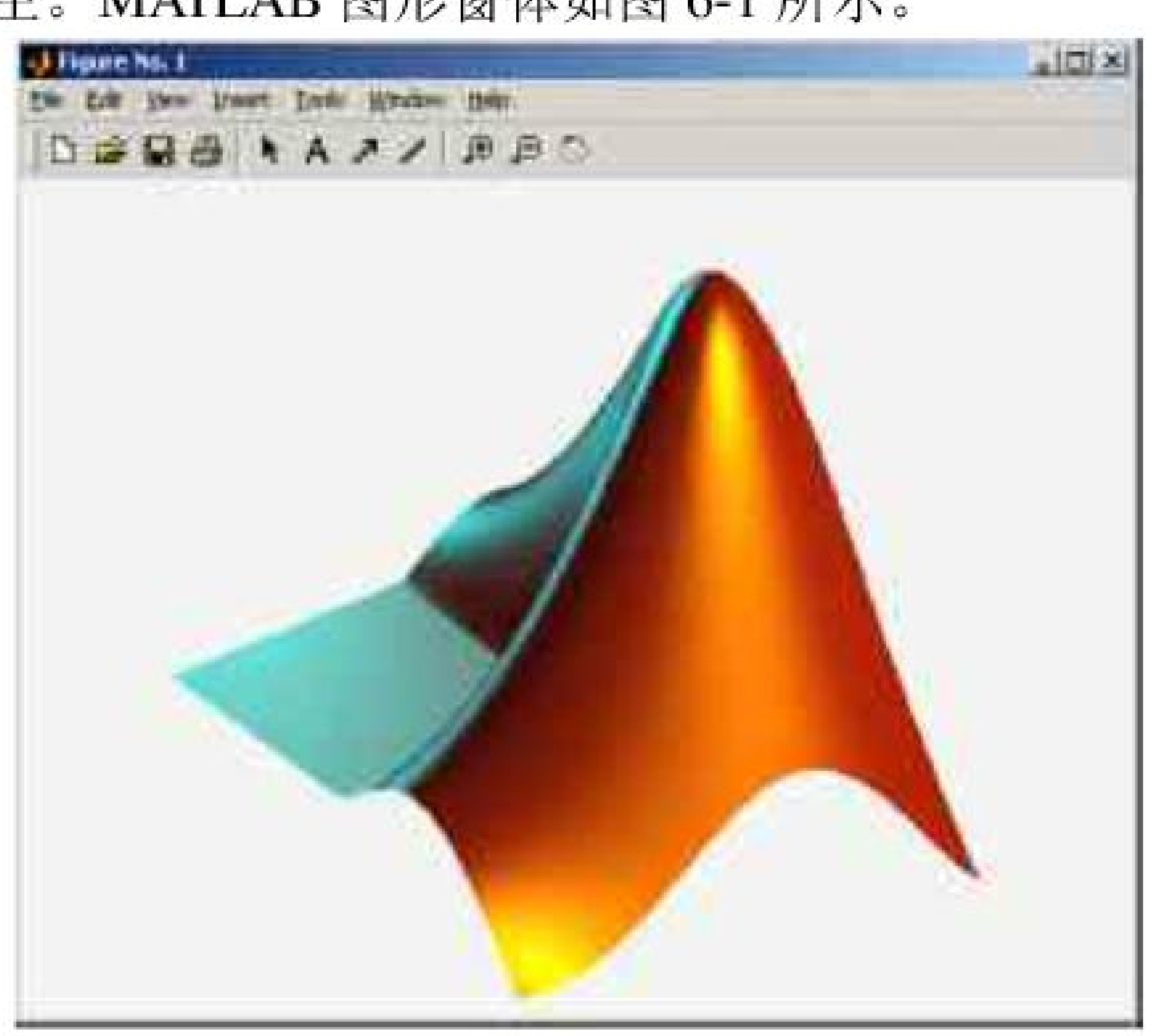


图 6-1　MATLAB 的图形窗体

MATLAB 的图形窗体主要包括如下几个部分：

■ 菜单栏：MATLAB 的图形窗体一般包括一个菜单栏，利用这个菜单栏可以完成对窗体中各种对象的基本操作，例如图形的打印导出等。

■ 工具条：图形窗体的工具条用来完成对图形对象的一般性操作，例如新建、打开、保存和打印，还有对图形窗体的编辑也是通过该工具条完成的。

■ 绘图区域：图形窗体的绘图区域是面积最大的一部分，在图 6-1 中就是绘制了 MATLAB 标志的矩形区域。在这个区域中可以绘制各种曲线，显示图形图像文件，以及完成对图形图像或者曲线的编辑。

一般地，在 MATLAB 中进行数据可视化的过程主要有如下步骤：

■ 准备需要绘制在 MATLAB 图形窗体中的数据。

■ 创建图形窗体，并且选择绘制数据的区域。一个 MATLAB 图形窗体可以包含多个绘图区域。

■ 使用 MATLAB 的绘图函数绘制图形或者曲线。

■ 设置曲线的属性，例如线型、线宽等。

■ 设置绘图区域的属性，并且添加数据网格线。

■ 为绘制的图形添加标题、轴标签或者标注文本等。

■ 打印或者导出图形。

在本章，将详细介绍绘制图形的上述过程，以及在不同过程中需要使用的各种函数。同时还将介绍在 MATLAB 图形窗体中进行简单数据处理统计的方法，这部分功能不仅利用了图形功能，还利用了 MATLAB 数据处理、科学计算的部分函数。创建图形用户界面的基本方法将在本教程的第七章进行介绍，而有关 MATLAB 高级的图形应用——句柄图形和图形用户界面的高级内容请读者参阅 MATLAB 的帮助文档。

6.2 基本二维绘图

MATLAB 中有各种数据类型——实数类型向量或矩阵、复数类型的向量或者矩阵。MATLAB 都可以将这些数据绘制在图形窗体中，并且能够使用不同的样式和颜色的线条表示不同的数据。MATLAB 将数据绘制在称之为轴(Axes)的图形对象中，同一个图形窗体中可以包含多个轴，也可以包含多个图形绘制区域(子图)。本小节将详细介绍这些基本的二维绘图指令和相应的使用方法。

6.2.1 基本绘图指令

在 MATLAB 中进行数据可视化使用最频繁的绘制函数就是 plot 函数，该函数能够将向量或者矩阵中的数据绘制在图形窗体中，并且可以指定不同的线型和色彩。同一个 plot 函数不仅能够绘制一条曲线，还可以一次绘制多条曲线。

plot 函数的基本使用语法格式为

绘制一条曲线：plot(xdata, ydata, 'color_linestyle_marker')

绘制多条曲线：plot(xdata1, ydata1, 'clm1', xdata2, ydata2, 'clm2', ……)

若在绘制曲线的时候没有指定曲线的色彩、线型和标识符，则 MATLAB 使用默认的设置。例子 6-1 中使用基本绘图指令绘制了曲线。

例子 6-1　MATLAB 基本绘图指令的使用。

在 MATLAB 命令行窗口中键入下面的指令：

```
>> x = 0:pi/1000:2*pi;
>> y = sin(2*x+pi/4);
>> plot(x,y)
```

例子 6-1 共有三条指令，前面两条是准备绘制的数据，x 和 y 两个变量为长度相同的行向量，其中 y 是利用三角函数处理的数据。而 plot 函数使用默认的设置将数据 x 和 y 绘制在图形窗体中。系统默认的设置为蓝色的连续线条。绘制的图形如图 6-2 所示。

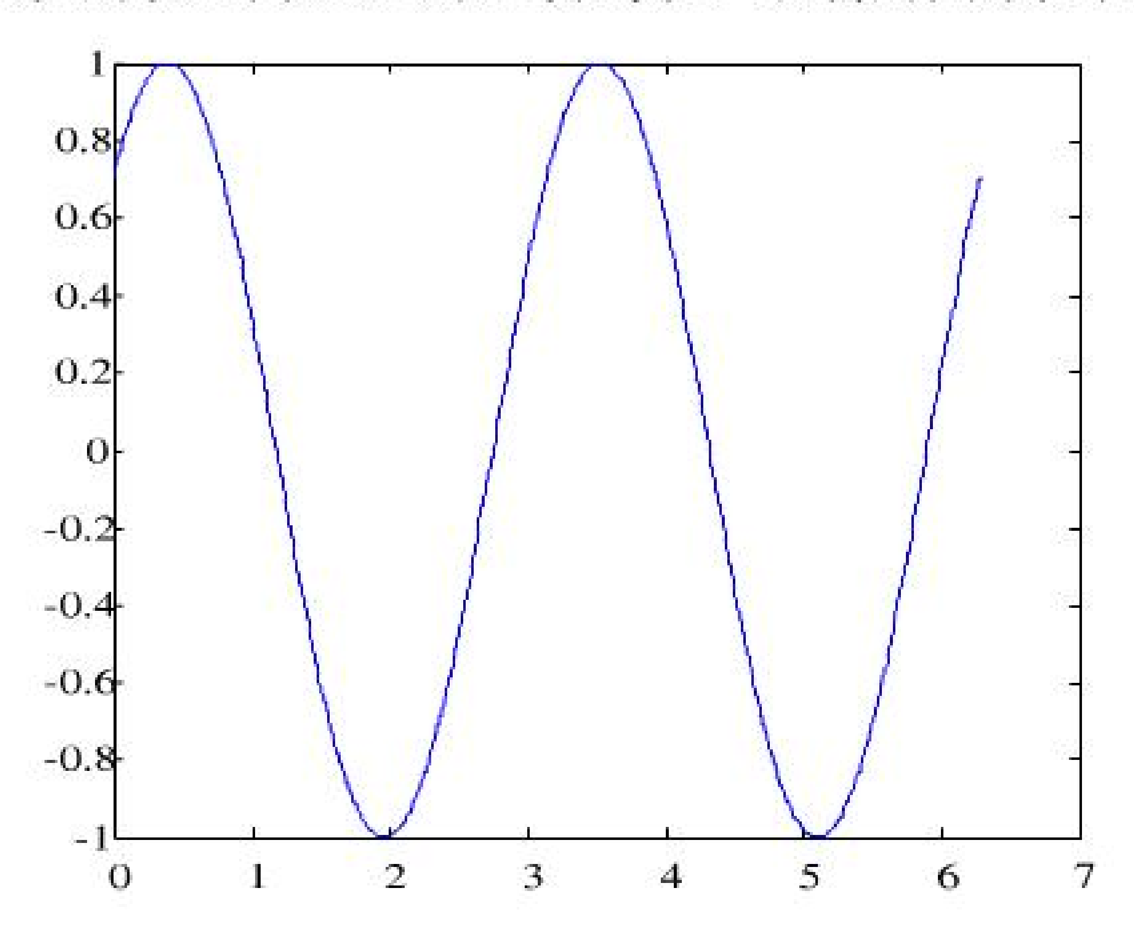


图 6-2　在 MATLAB 图形窗体中绘制蓝色曲线

plot 函数能够同时绘制多条曲线，在 MATLAB 命令行窗口中，键入下面的指令：

```
(继续前面的指令)
>> plot(x,y,x,y+1,x,y+2)
```

这时绘制的图形如图 6-3 所示。

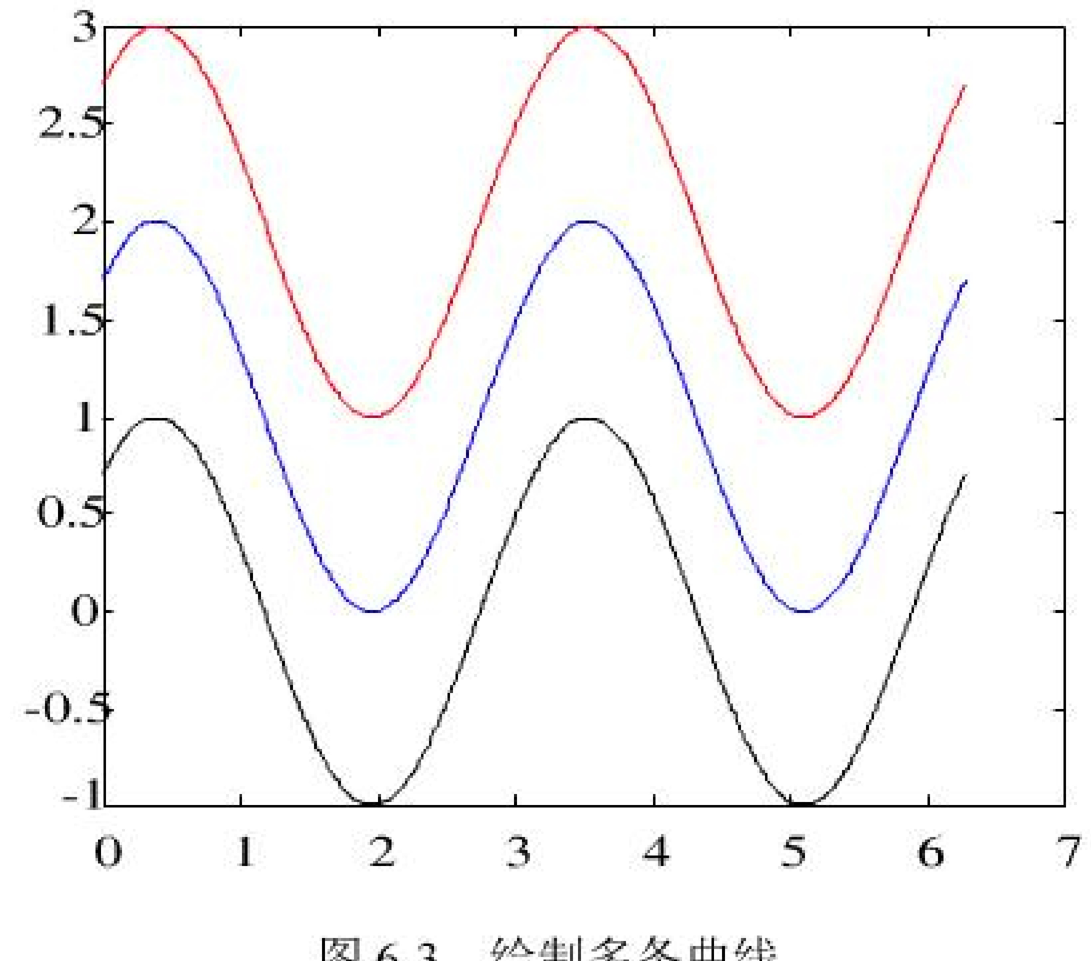


图 6-3　绘制多条曲线

在图形窗体中，由下至上分别为绘制的第一、二、三条曲线，根据系统的默认设置分别为蓝色、绿色和红色。

例子 6-1 说明了 plot 函数的基本用法，同时也说明了 plot 函数的系统默认设置。不过例子中使用的数据是两个向量，分别作为 X 轴的数据和 Y 轴的数据。那么对于 MATLAB 是如何处理的呢？

利用 plot 函数可以直接将矩阵的数据绘制在图形窗体中，这个时候 plot 函数将矩阵的每一列数据作为一条曲线绘制在窗体中，如例子 6-2 所示。

例子 6-2　利用 **plot** 函数绘制矩阵数据。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> A = pascal(5)
A =
      1     1     1     1     1
      1     2     3     4     5
      1     3     6    10    15
      1     4    10    20    35
      1     5    15    35    70
>> plot(A)
```

这时得到的图形如图 6-4 所示。

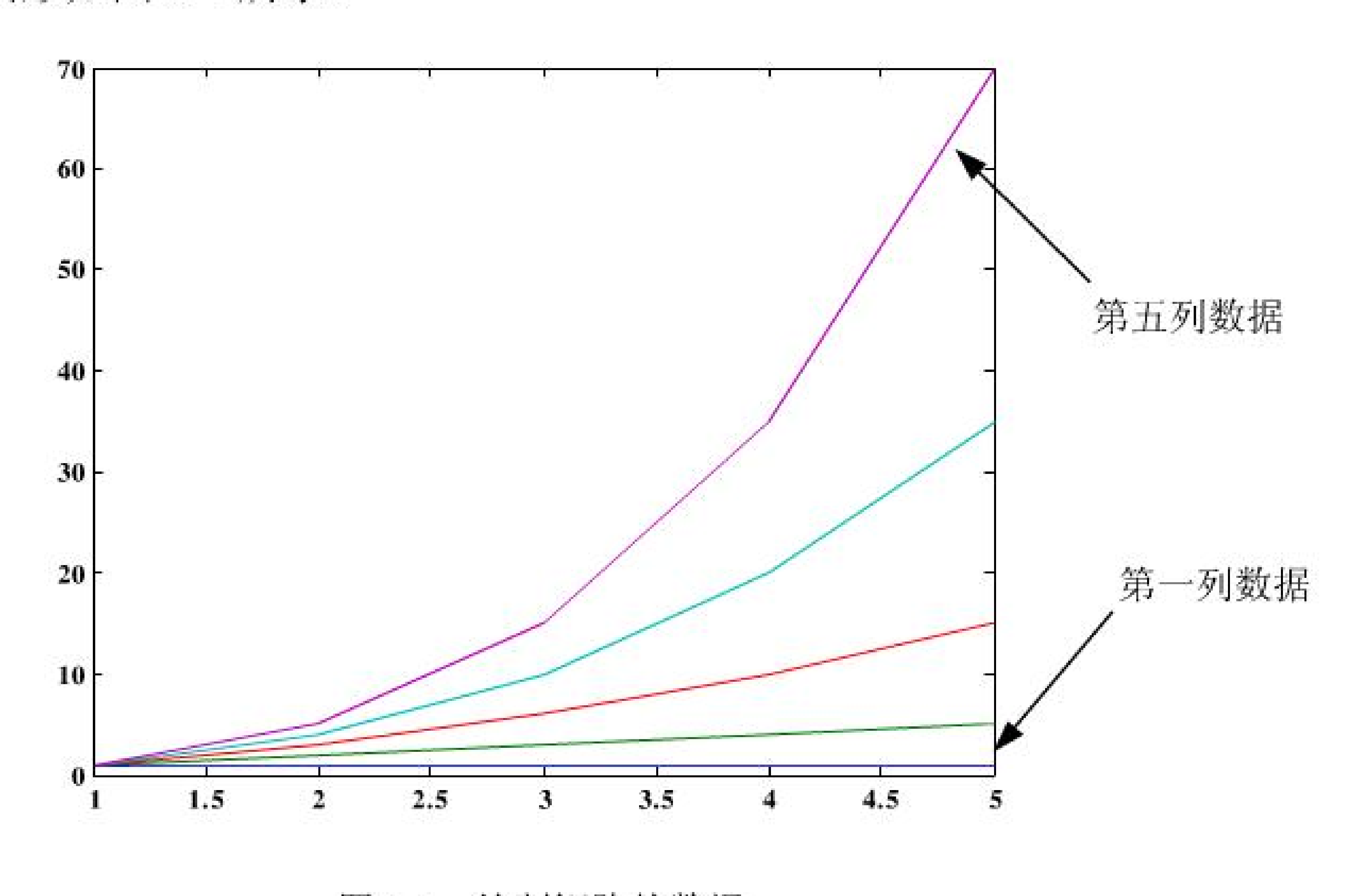


图 6-4　绘制矩阵的数据

绘制矩阵的数据时，plot 函数只要直接将需要绘制的矩阵作为参数输入就可以了，这时 MATLAB 自动地将矩阵中每一列数据作为一条曲线绘制出来，例如在图 6-4 中，就包含了五条曲线。

6.2.2　设置曲线的样式属性

为了能够在 plot 函数中控制曲线的样式，MATLAB 预先设置了不同的曲线样式属性值，

分别控制曲线的色彩、线型和标识符，在表 6-1 中对 plot 函数的标识符进行了总结。

表 **6-1 plot** 函数的标识符

色彩 color	说明	时标 m r r	说明	线型 l l	说明
r	红色		加号		实线
	绿色	o	圆圈		虚线
	蓝色		星号	:	点线
c	青		点	。	点划线
m	洋红		十字		
	黄色		矩形		
	黑色		菱形		
	白色		上三角		
			下三角		
			右三角		
			左三角		
			五边形		
			六边形		

将表 6-1 中的各种标识符组合起来就可以控制绘制曲线的特性了，参见例子 6-3 的说明。

例子 6-3 设置曲线的样式。

在 MATLAB 命令行窗口中，键入下面的指令：

lo r m

例子 6-3 在同一个图形窗体中绘制三条不同的曲线，为了区分这些曲线，使用了不同的时标、色彩和线型，绘制的曲线如图 6-5 所示。

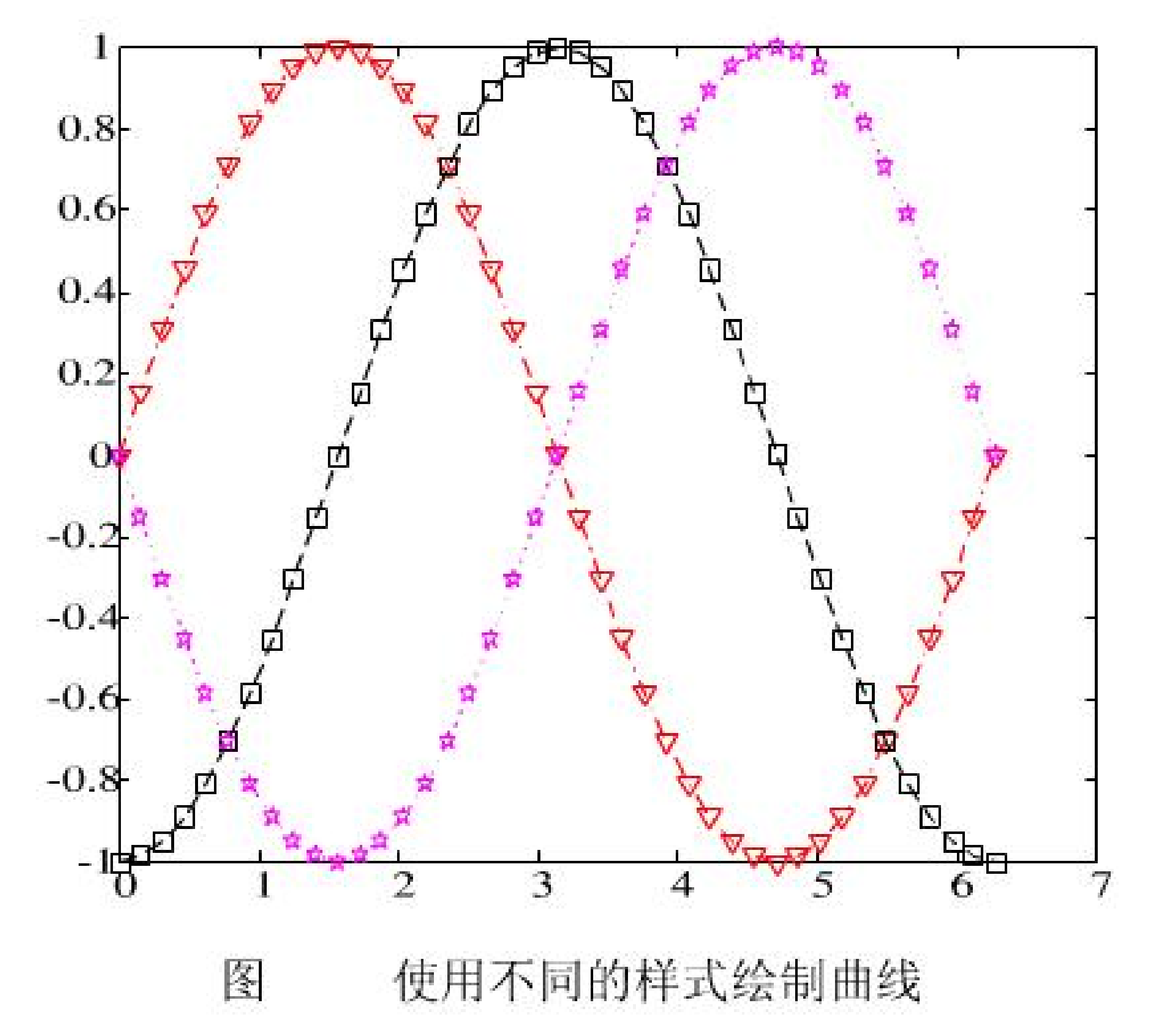


图 使用不同的样式绘制曲线

提示：常用的绘图命令

为了能够更加直观地观察数据曲线，可以使用 grid 命令将轴的坐标线绘制出来，具体的做法为 grid on。

执行该命令后，图形窗体的轴将显示坐标网格线，图 6-6 就是将例子 6-3 的结果添加上坐标网格线之后的效果。

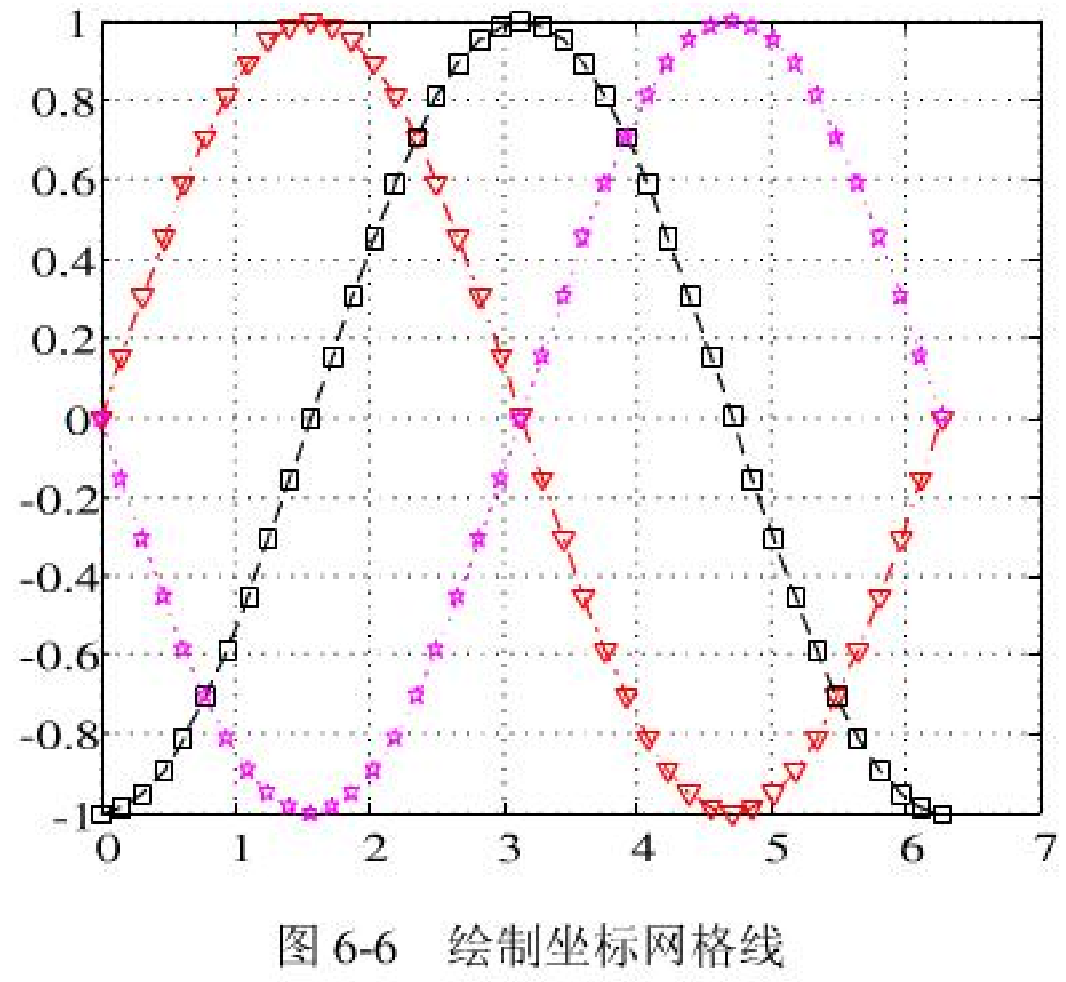


图 6-6　绘制坐标网格线

若不希望出现坐标网格线，则可以执行 grid off 命令。执行 grid 命令，图形窗体的轴将在有网格线和无网格线之间切换。

若需要向已经存在曲线的图形窗体中增加曲线，则可以使用 hold on 命令锁定当前的图形窗体，之后所有绘图操作的结果都会显示在当前的图形窗体中。使用 hold off 命令则解除锁定状态，这时候任何绘图操作都将清除当前图形窗体已经绘制的内容。单纯使用 hold 命令则将在锁定与非锁定状态之间切换。

清除当前图形窗体内容的指令为 clf。创建新的图形窗体的指令为 figure。

若不希望将绘制的曲线点连接起来，则在使用 plot 函数绘制曲线的时候不要指定线条的样式，仅指定时标选项和色彩选项，这时就不会将绘制的点用线连接起来了。例如接例子 6-3 执行下面的指令：

```
>> plot(t,y,'r ',t,y ,' ',t,y ,' p'
>> r o
```

得到的图形如图 6-7 所示。

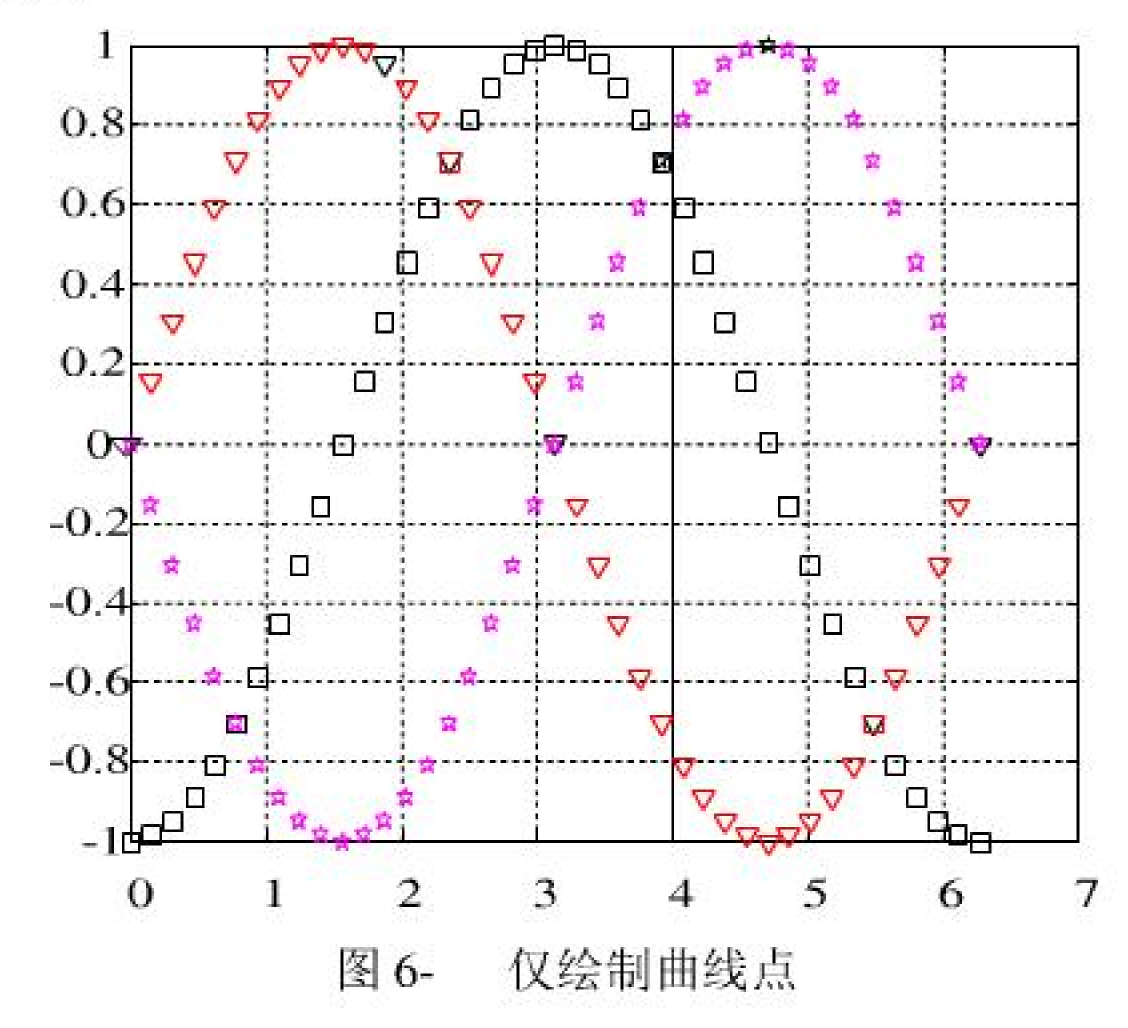


图 6-　仅绘制曲线点

MATLAB 还允许对利用 plot 函数绘制的曲线进行更细致的控制，不过需要通过设置曲线的属性来完成。MATLAB 图形对象都有自己的属性，通过修改属性就可以修改图形曲线的外观，这也是句柄图形和图形用户界面操作图形对象的方法。绘制曲线时，可以通过修改下列属性完成对曲线细节的设置：

- LineWidth：曲线的宽度，单位为 point。
- MarkerEdgeColor：曲线时标边缘的色彩。
- MarkerFaceColor：填充曲线时标的色彩。
- MarkerSize：曲线时标的大小，单位为 point。

例子 6-4　设置曲线的细节属性。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> x = -pi:pi/10:pi;
>> y = tan(sin(x)) - sin(tan(x));
>> plot(x,y,'--rs','LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10)
```

例子 6-4 中设置了曲线的线宽、Marker 的填充色、边缘色等属性，于是得到的绘图结果如图 6-8 所示。

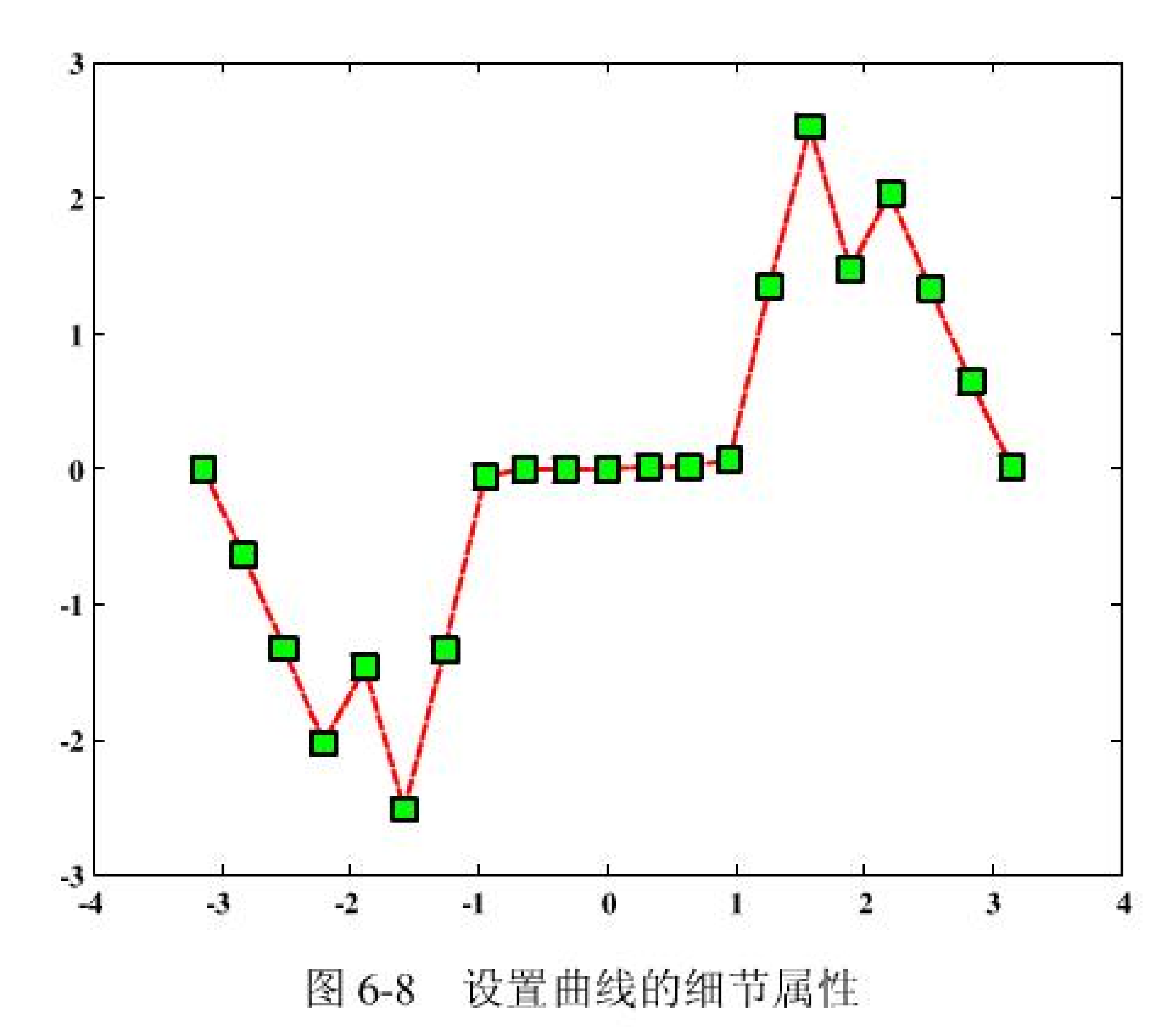


图 6-8　设置曲线的细节属性

有关更详细的属性设置请参阅 MATLAB 的帮助文档。

6.2.3 使用子图

MATLAB 的图形窗体中不仅可以包含一个轴，还可以划分为多个图形显示区域，每个图形显示区域彼此独立，用户可以根据自己的需要把数据绘制在指定的区域中，这种特性就是利用 MATLAB 图形窗体的子图功能来完成的。使用子图的方法非常简单，只要使用 subplot 函数选择绘制区域即可。

subplot 函数把现有的图形窗体分割成指定行数和列数的区域，在每个区域内都可以包含一个绘图轴，利用该函数选择不同的绘图区，然后所有的绘图操作都将结果输出到指定的绘图区中。

subplot 函数的基本用法如下：

```
subplot(m,n,p)
```

其中，m 和 n 为将图形窗体分割成的行数和列数，p 为选定的窗体区域的序号，以行元素优先顺序排列。

例如，在 MATLAB 命令行窗口中键入指令：

```
>> subplot(2,3,4)
```

则 MATLAB 将图形窗体分割成为二行三列，并且将第四个绘图区域设置为当前的绘图区域。例子 6-5 说明了子图的使用方法。

例子 6-5　使用子图——subplotex.m。

```
function subplotex
% 子图的使用示例
x = 0:.1:2*pi;
% 创建新的图形窗体
figure(1);clf;
% 分隔窗体为 2 行 2 列，分别在不同的区域绘图
subplot(2,2,1);plot(1:10);grid on;
subplot(2,2,2);plot(x,sin(x));grid on;
subplot(2,2,3);plot(x,exp(-x),'r');grid on;
subplot(2,2,4);plot(peaks);grid on;
% 子图的使用特别的用法
% 创建新的图形窗体
figure(2);clf
% 图形窗体分割为 4 行 5 列，选择第 2~4 号区域
subplot(4,5,2:4);plot(1:10);grid on;
% 选择向量中指定的区域
subplot(4,5,[7 8 9 12 13 14]);plot(peaks);grid on;
% 选择单一的区域
subplot(4,5,11);plot(membrane);grid on;
% 选择多个区域
subplot(4,5,16:20);surf(membrane);grid on;
```

注意在例子 6-5 中，选择多个绘图区域时使用的 subplot 函数的格式。

运行例子 6-5 的代码，在 MATLAB 命令行窗口中，键入下面的指令：

```
>> subplotex
```

则 MATLAB 创建两个图形窗体，分别绘制分割的图形，如图 6-9、6-10 所示。

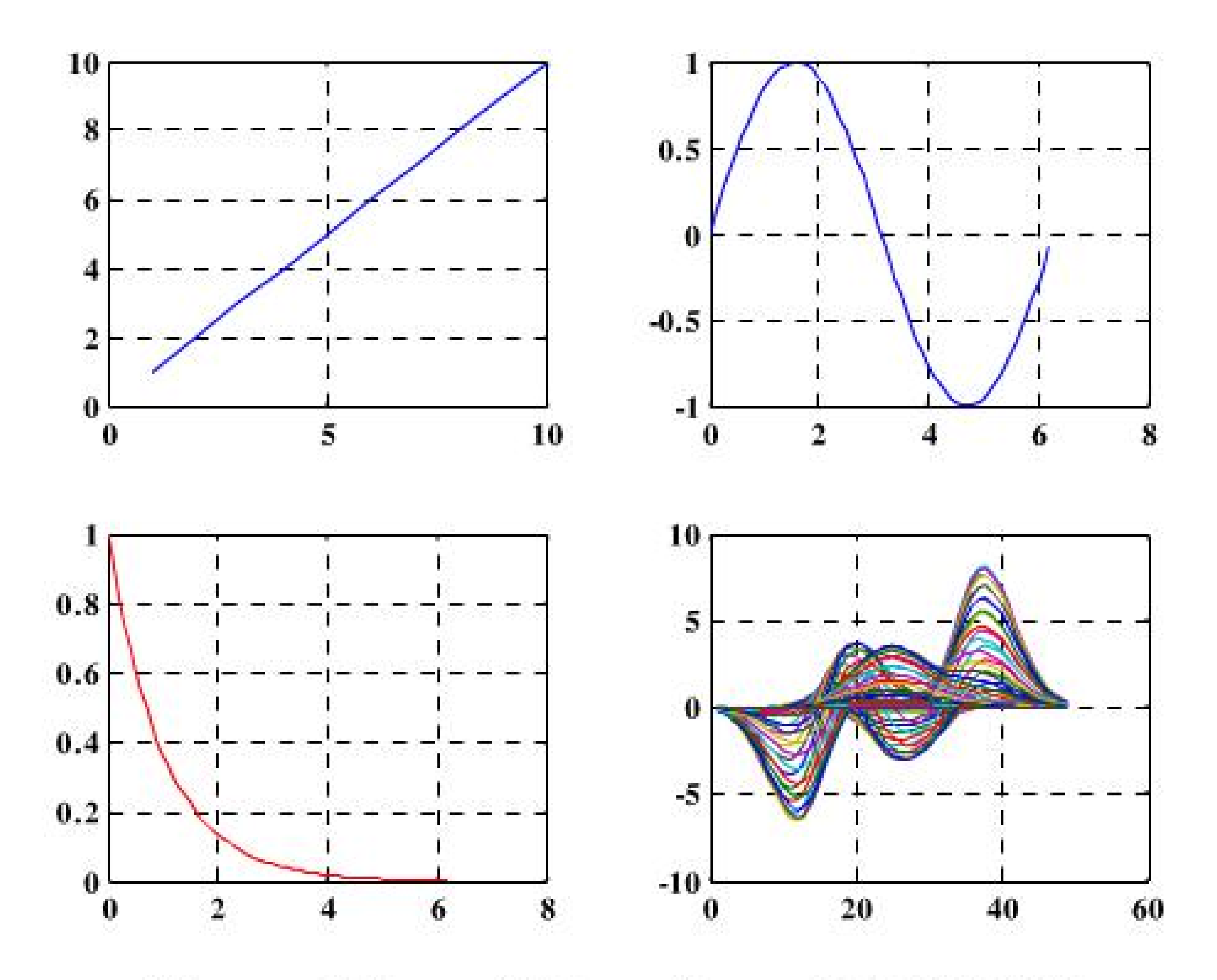


图 6-9　例子 6-5 代码 007 行~010 行的图形结果

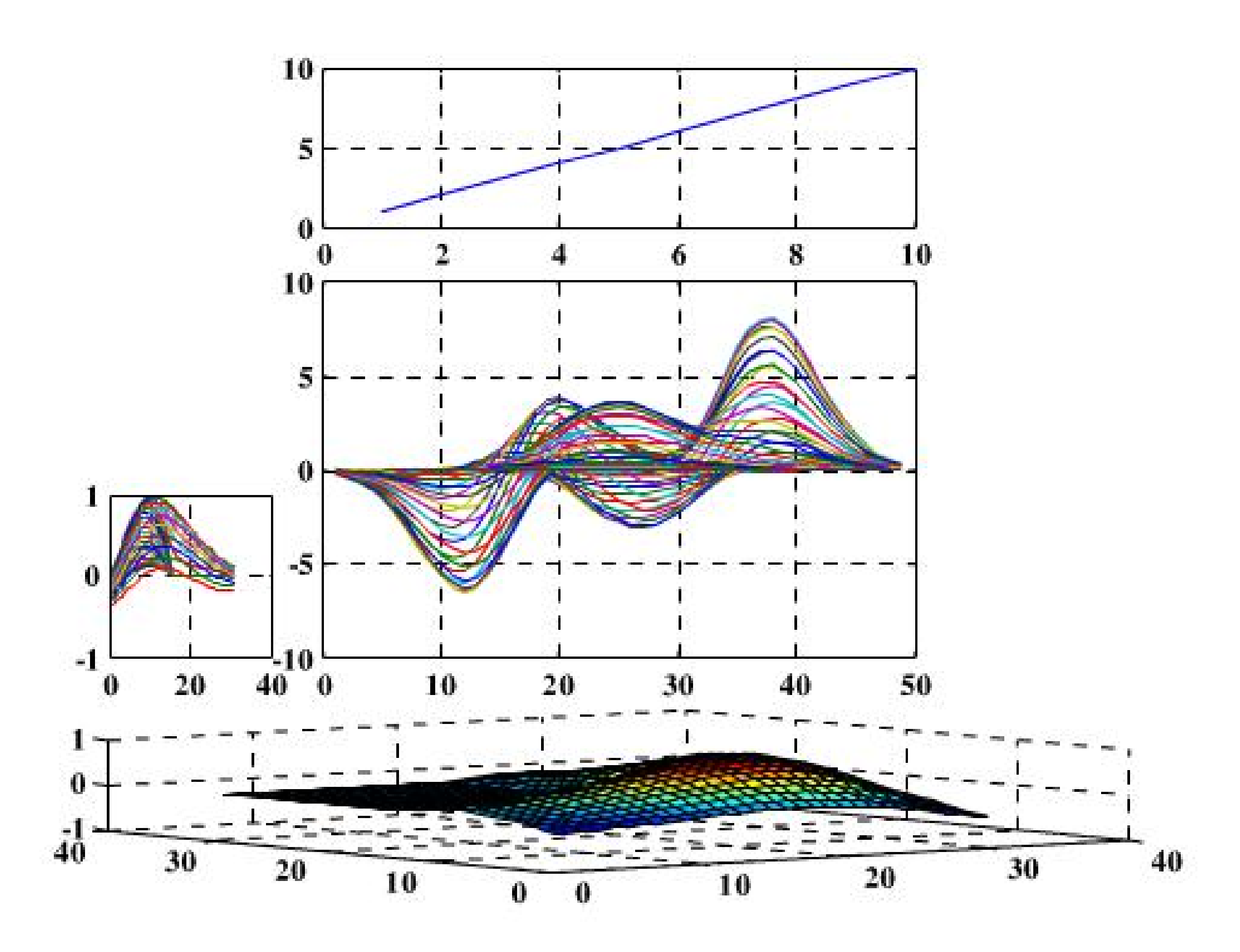


图 6-10　例子 6-5 代码 014 行~021 行的图形结果

使用 subplot 函数选择不同的绘图区域能够得到不同的结果。在例子 6-5 的代码中，第二个图形窗体每个轴都不相同，这种结果是利用了 subplot 函数选择多个绘图区域完成的。请读者根据例子的代码和运行结果察看并学习 subplot 函数的使用方法。

6.2.4　控制绘图区域

所谓 MATLAB 的绘图区域是指图形窗体中的轴(Axes)，需要牢记一点，所有 MATLAB 的图形对象都是绘制在轴的上面，所以控制绘图的区域也就是控制轴的显示区域。利用 MATLAB 的图形功能绘制图形时，MATLAB 自动地根据绘制的数据调整轴的显示范围，它能够保证将所遇的数据以适当的比例显示在轴中。用户同样可以修改轴显示的范围，而且还可以修改轴的标注，修改这些特性需要使用 axis 函数，并且设置相应的属性。

axis 函数可以修改图形窗体轴的范围，它的基本语法格式如下：

```
axis([xmin xmax ymin ymax])
```

其中，xmin 和 xmax 决定 X 轴的显示范围，ymin 和 ymax 决定 Y 轴的显示范围。

若在 MATLAB 命令行窗口中，直接键入下面的指令：

```
>> axis
ans =
     0     1     0     1
```

则 MATLAB 按照默认的设置自动创建一个图形窗体，包含一个空白的轴，其中 X 轴的范围和 Y 轴的范围都为 0~1。

例子 6-6　**axis** 函数使用示例。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> x = 0:pi/100:pi/2;
>> y = tan(x);
>> plot(x,y,'ko')
>> grid on
```

这时的图形窗体中的内容如图 6-11 所示。

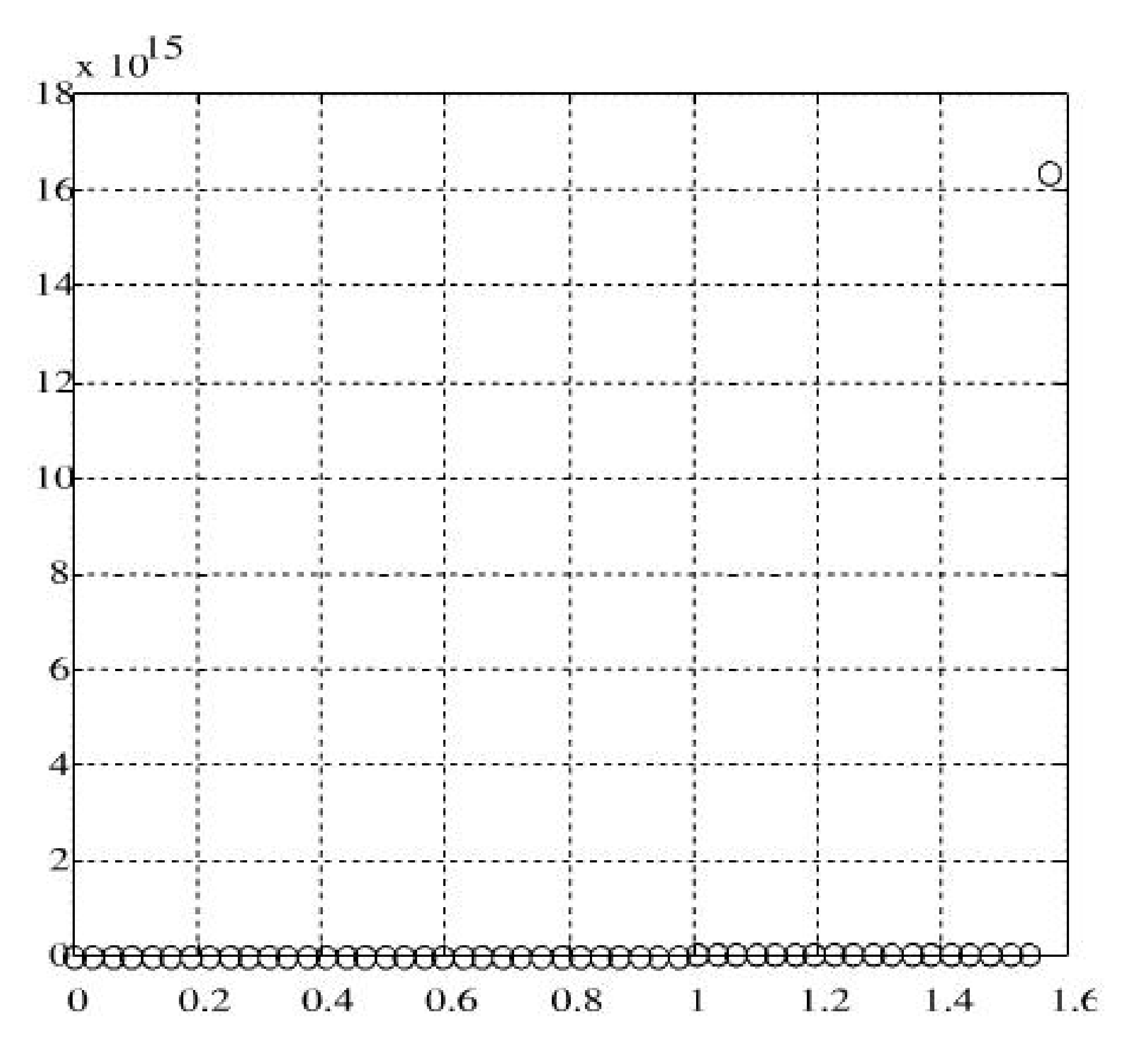


图 6-11　自动比例设置的图形窗体内容

可以看出，由于默认的图形窗体轴能够根据数据的范围自动调节图形显示的比例，所以图 6-11 显示的结果并不是那么直观，绘制的数据几乎排成了一条直线，所以需要修改显示范围。

```
>> axis([0,pi/2,0,5])
```

该命令将图形窗体轴的范围缩小，这时，前面数据的细节就可以很容易地查看出来了，如图 6-12 所示。

axis 函数除了能够用来直接设置轴的范围外，还能够用来设置轴的行为，例如设置轴是否按照数据的范围自动调节等，具体的用法请参阅 MATLAB 的帮助文档或者在线帮助。

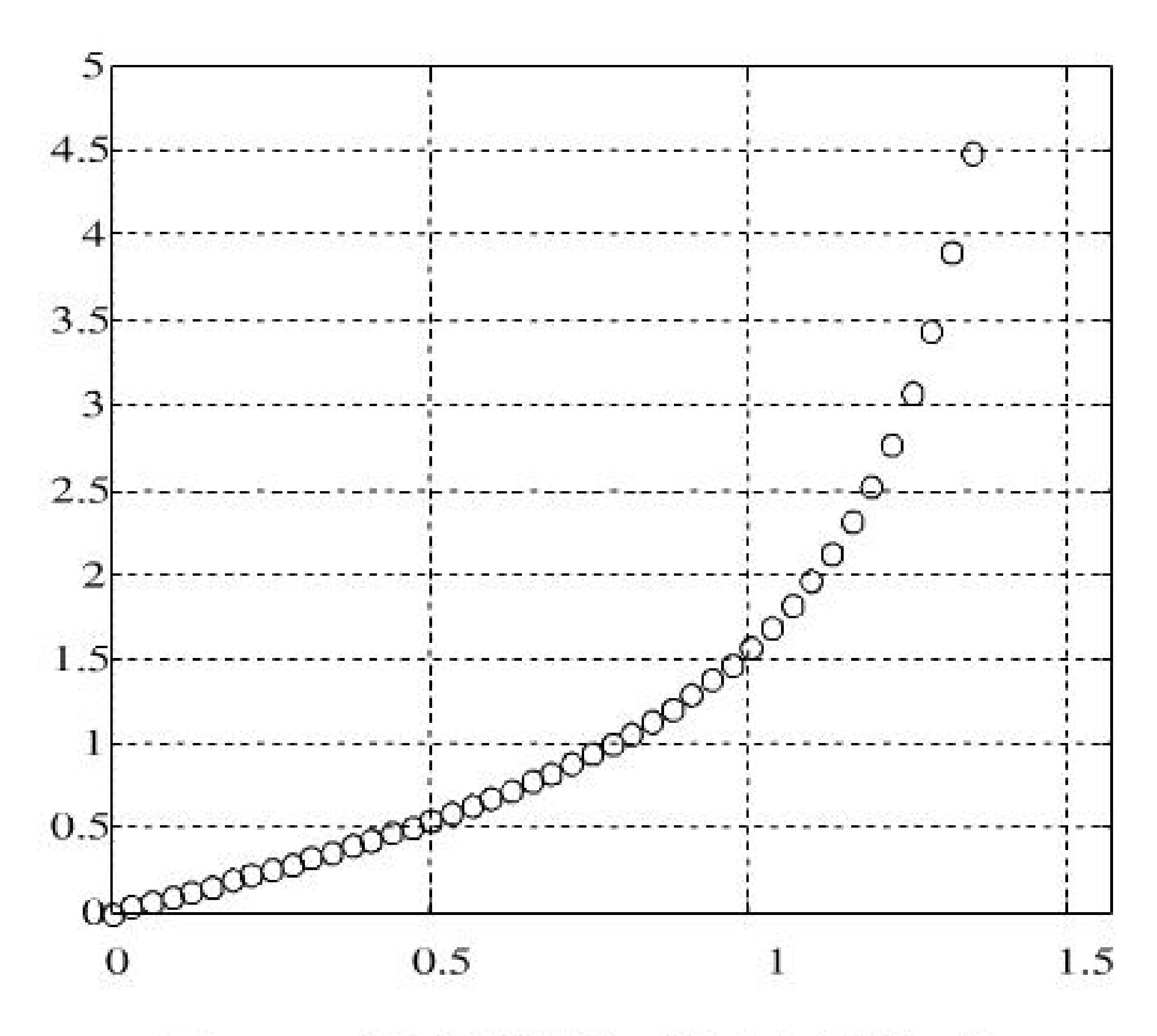


图 6-12　设置轴的属性以查看数据的细节

绘制曲线的轴还需要设置坐标线之间的间隔(Ticks)，在默认的情况下，MATLAB 按照绘制数据的范围设置坐标线的间隔，这种间隔的设置是自动完成的，用户也可以根据自己的需要设置这些间隔，具体的方法是通过设置轴的 XTick 或者 YTick 属性实现对 X 轴或者 Y 轴的坐标间隔设置，这里举例说明。

例子 6-7　设置轴的坐标间隔。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> x = -pi:pi/10:pi;
>> y = cos(x);
>> plot(x,y,'-r^');
>> grid on
```

这时 MATLAB 图形窗体如图 6-13 所示。

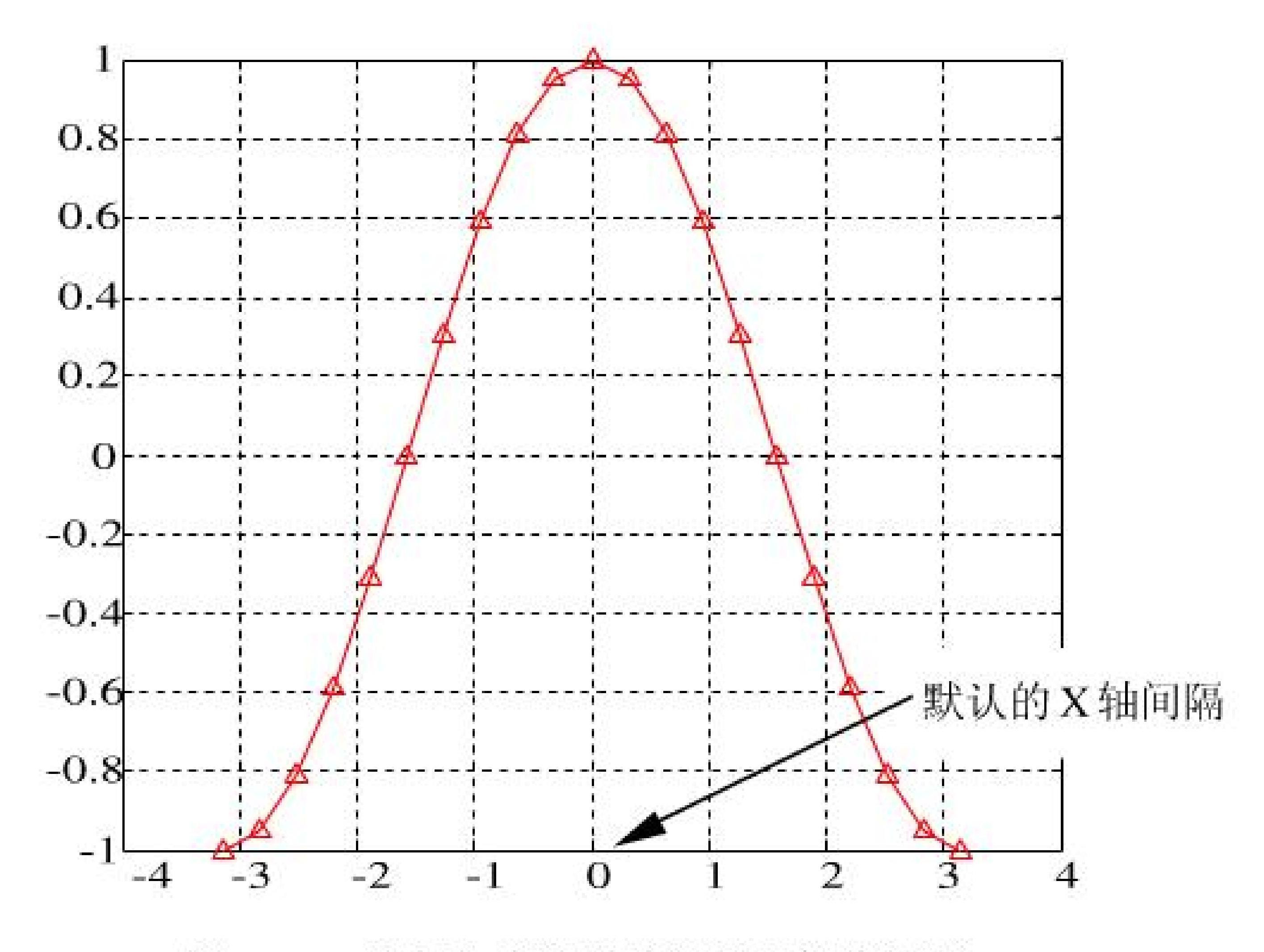


图 6-13　绘制曲线使用默认的坐标线间隔

修改轴的坐标间隔属性，需要使用 set 函数修改指定对象的 XTick 或者 YTick 属性值。在本例子中，修改 X 轴的范围和坐标间隔：

```
>> axis([- pi,pi, - inf,inf])
>> set(gca,'XTick', - pi:pi/4:pi)
```

这时 MATLAB 的图形窗体如图 6-14 所示。

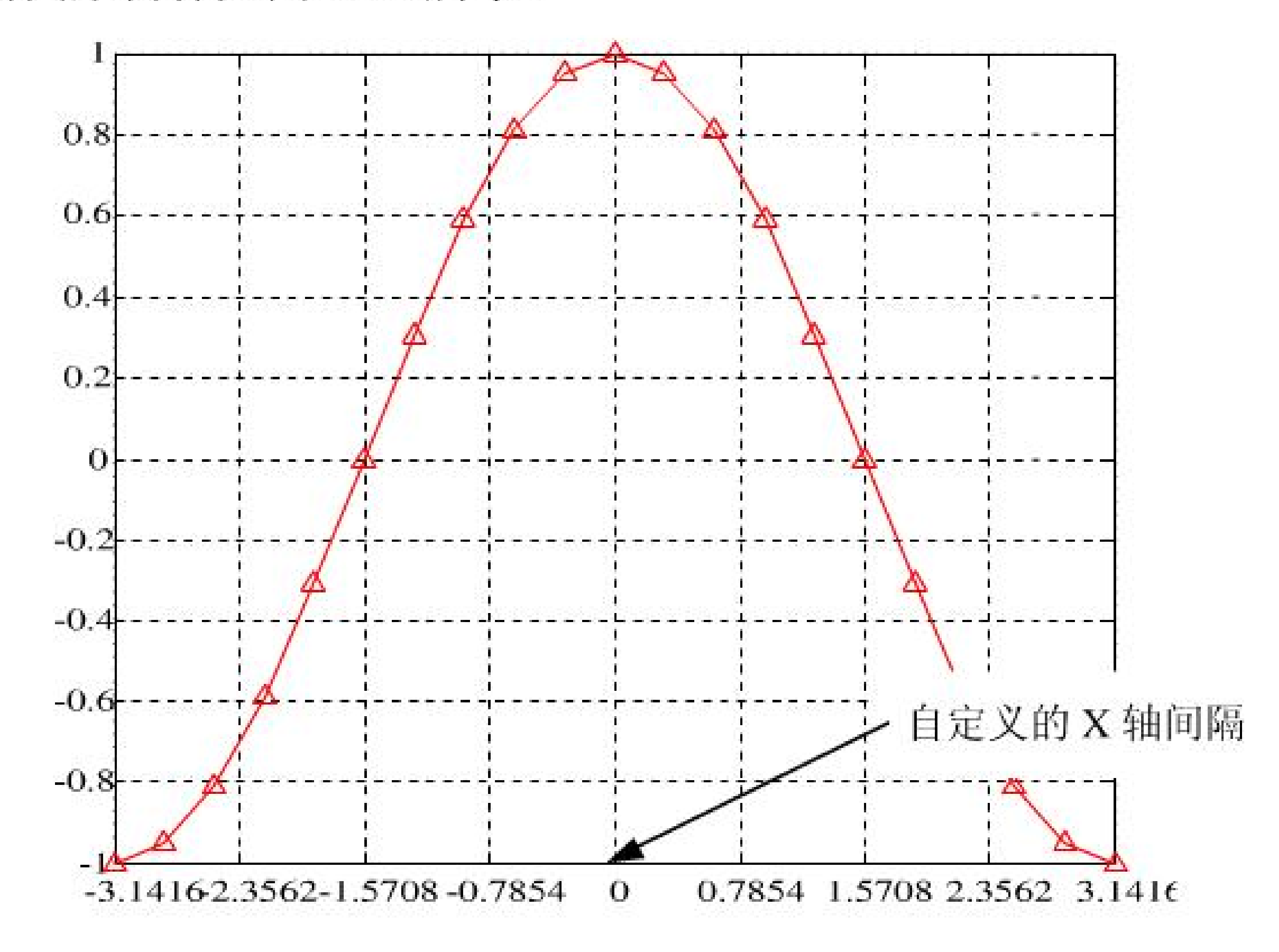


图 6-14　设置坐标间隔线和坐标范围

注意：

在设置坐标轴的范围时，若将坐标轴取值设定为 inf(如例子 6-7)，则表示该坐标轴的范围为自动，也就是说，在本例子中 X 轴的范围为[-p,p]，则 Y 轴按照绘制数据的范围自动设定。

提示：

set 函数和 get 函数是用来设置/获取 MATLAB 图形对象属性的常用函数。MATLAB 的图形对象都包含有不同的属性和相应的属性数值，这些属性和属性值直接决定了 MATLAB 图形对象的表现形式。例如在例子 6-7 中设置了轴对象的属性——XTick，修改之后 X 轴的坐标间隔发生了变化。另外，本例子使用 gca 函数来获取当前的轴图形对象的句柄，有关图形对象的内容可以参阅 MATLAB 帮助文档，而 get 和 set 函数将在后面章节中详细介绍。

到目前，例子 6-7 图形的 X 轴已经基本符合要求了，但是 X 轴的标注依然是数字，而很多时候需要使用特殊数值表示，比如本例子中使用常数“pi”，这个时候需要修改轴的 XTickLabel 或者 YTickLabel 属性。

继续例子 6-7，在 MATLAB 命令行窗口中键入下面的指令：

```
>> label = {'- pi','','- pi/2','','0','','pi/2','','pi'}
label =
  Columns 1 through 6
    '-pi'      ''     '- pi/2'      ''      '0'      ''
  Columns 7 through 9
    'pi/2'      ''     'pi'
```

```
>> set(gca,'XTickLabel',label)
```

这时的图形窗体如图 6-15 所示。

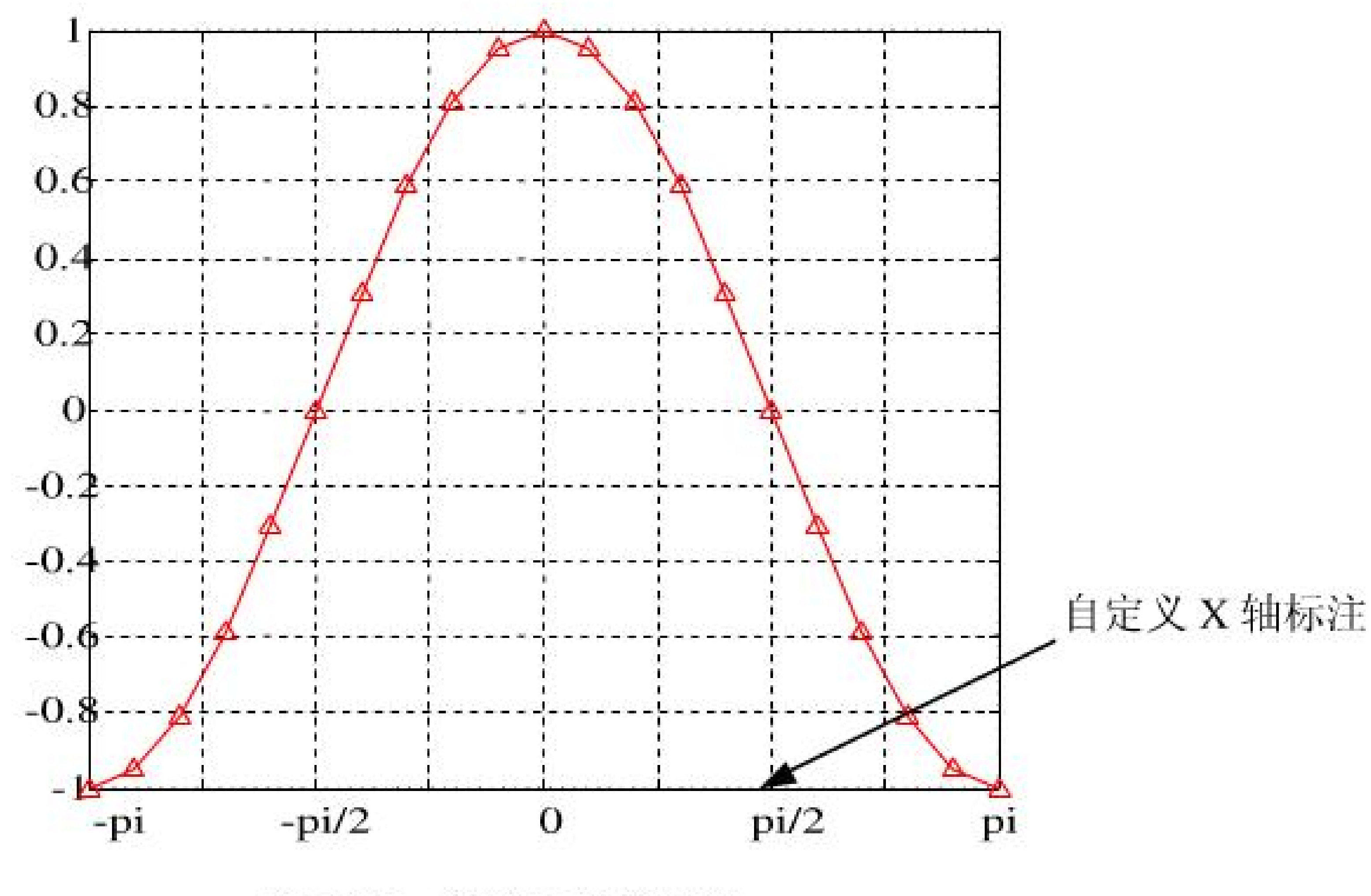


图 6-15　设置 X 轴的标注

若需要设置 Y 轴的坐标轴线和相应的标注可以参考例子 6-7 的方法完成相应的工作。利用 axis 函数还能够控制图形坐标系的其他特性，这里就不一一详细介绍了。请读者自己阅读 MATLAB 的帮助文档或者在线帮助。

6.2.5　图形编辑器

MATLAB 不仅能够显示数据和图形，而且还可以利用图形窗体中的工具对图形对象进行编辑，这时的图形窗体需要进入到编辑模式。在编辑模式下，可以向图形窗体中的对象添加文本、箭头、直线等，还可以利用 MATLAB 提供的编辑工具完成图形对象的编辑工作。

进入图形编辑模式有以下几种方法：

- 执行图形窗体中"Tool"菜单下的"Edit Plot"命令。
- 单击图形窗体工具栏中选择对象按钮。
- 执行"Edit"菜单下的菜单命令或者"Insert"菜单下的菜单命令都可以进入编辑模式。
- 在 MATLAB 命令行窗口中，键入"plotedit"指令。

进入图形编辑模式后，可以向图形添加各种元素，完成工作后，只要单击按钮就可以回到正常的显示模式。

图形编辑模式下比较重要的工作是设置各种图形对象的属性，这里以设置轴对象的属性为例来说明这一过程。

例子 6-8　利用图形编辑模式编辑轴对象属性。

本例子使用例子 6-7 的指令绘制基本的图形，请参阅例子 6-7 和图 6-13。

首先进入图形编辑模式，利用前面介绍的不同方法都能够进入编辑模式，进入编辑模式后，用鼠标单击轴，这时图形窗口如图 6-16 所示。

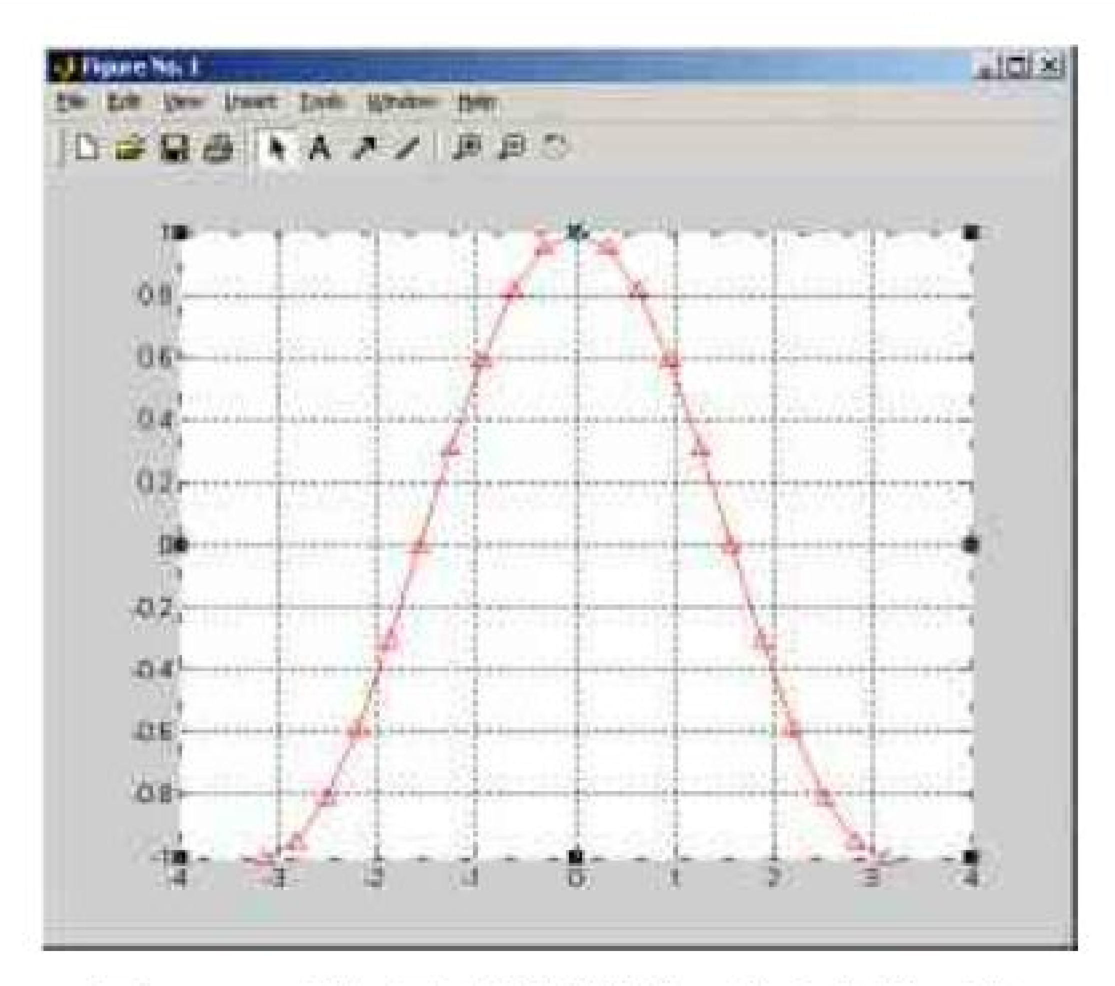


图 6-16　进入图形编辑模式，并选择轴对象

这时可以利用不同的方式打开轴对象的属性编辑器：

■ 执行“Edit”菜单下的“Axes Properties”命令。
■ 执行鼠标右键快捷菜单下的“Properties”命令。
■ 在 MATLAB 命令行窗口中键入指令 propedit，在弹出的图形属性对话框中选择轴对象。

轴对象的属性编辑器如图 6-17 所示。



图 6-17　轴属性编辑器

在轴属性编辑器中，可以设置 X、Y 和 Z 轴的属性，以及有关绘图的其他属性。这里需要将 X 轴的属性设置成例子 6-7 的数值。需要修改的地方包括 Limit、Tick 和 Labels。

修改完毕，单击“Apply”按钮就可以得到结果，如图 6-18 所示。

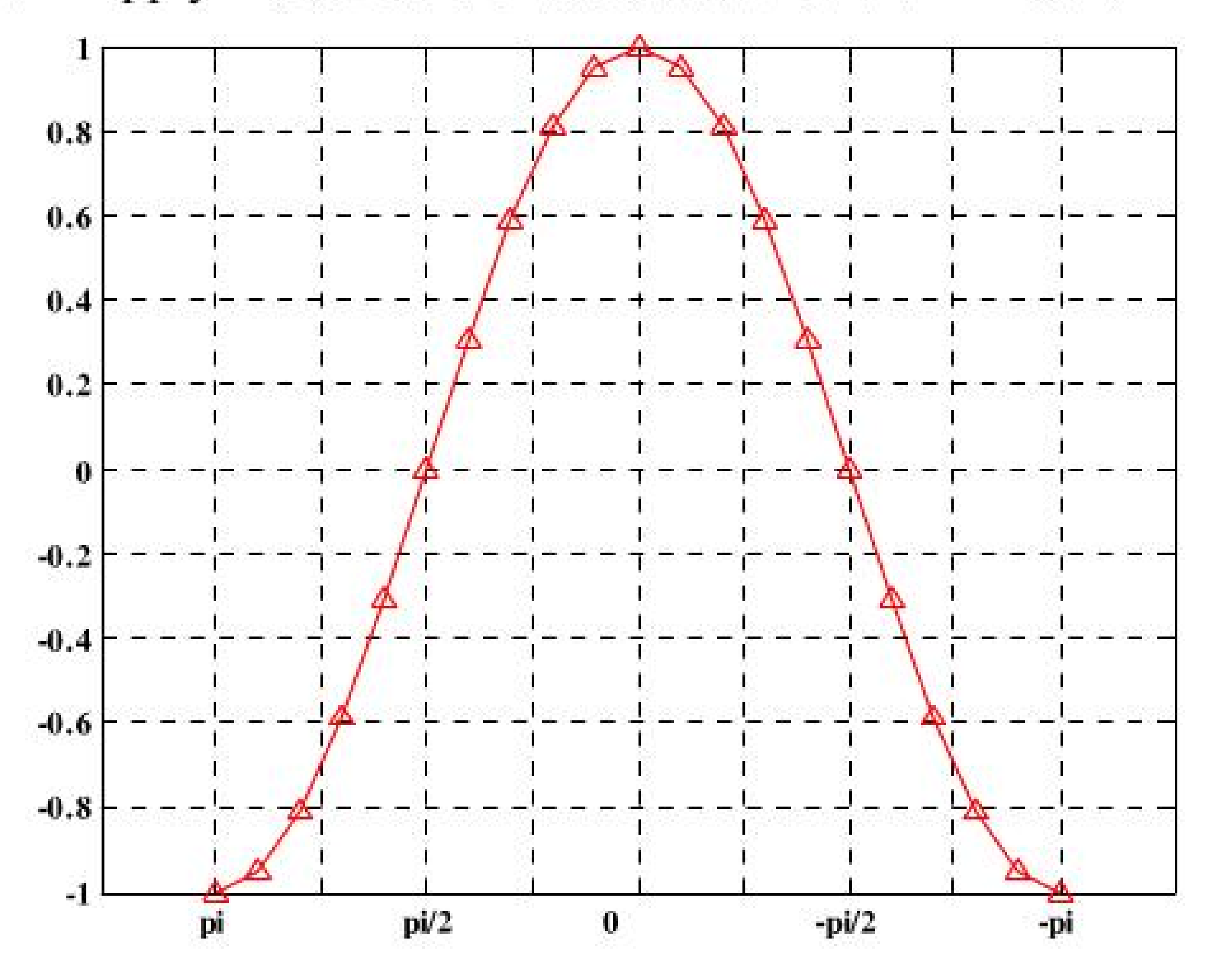


图 6-18　设置轴对象属性得到的结果

注意：

在设置 Tick 属性时，输入到文本框中的内容为-pi:pi/4:pi，然后 MATLAB 会自动计算得到相应的数值，所以在属性编辑对话框中看到了实际的数值向量。

在图形编辑模式下可以将图形窗体中的对象直接删除、复制和粘贴。注意，这里的删除操作无法回退，也就是说图形对象的删除操作不能恢复。而复制和粘贴图形对象的时候，首先选择需要复制的图形对象，然后利用快捷键 Ctrl+C 将图形对象复制，接着新建一个图形窗体，在图形窗体的编辑模式下，利用快捷键 Ctrl+V 将图形粘贴到新的窗体中。注意，复制和粘贴图形对象的过程一定要在图形编辑模式下才能够完成。

不仅轴对象具有属性编辑器，MATLAB 的各种图形对象都具有自己的属性编辑器对话框，利用属性编辑器能够编辑图形窗体中各种对象的相关属性，具体的操作请参阅 MATLAB 的帮助文档，也请有兴趣的读者自己动手实践一下，以便更好地掌握这些特性。

6.3　格 式 化 绘 图

所谓格式化绘图是指在 MATLAB 的图形窗体中，为 MATLAB 的图形对象添加必要的注释、标题或者其他文本信息，让 MATLAB 的图形能够表述更加丰富的信息。MATLAB 通过一系列的函数完成这些格式化绘图的功能。本小节将详细介绍在 MATLAB 图形窗体中添加图形标题、文本注释、轴标签等格式化图形信息的方法。

6.3.1　增加文本信息

MATLAB 图形窗体的文本信息主要包括图形标题、文本注释、轴标签和图例等，图 6-19 中的 MATLAB 图形窗体包含了所有这些文本信息。

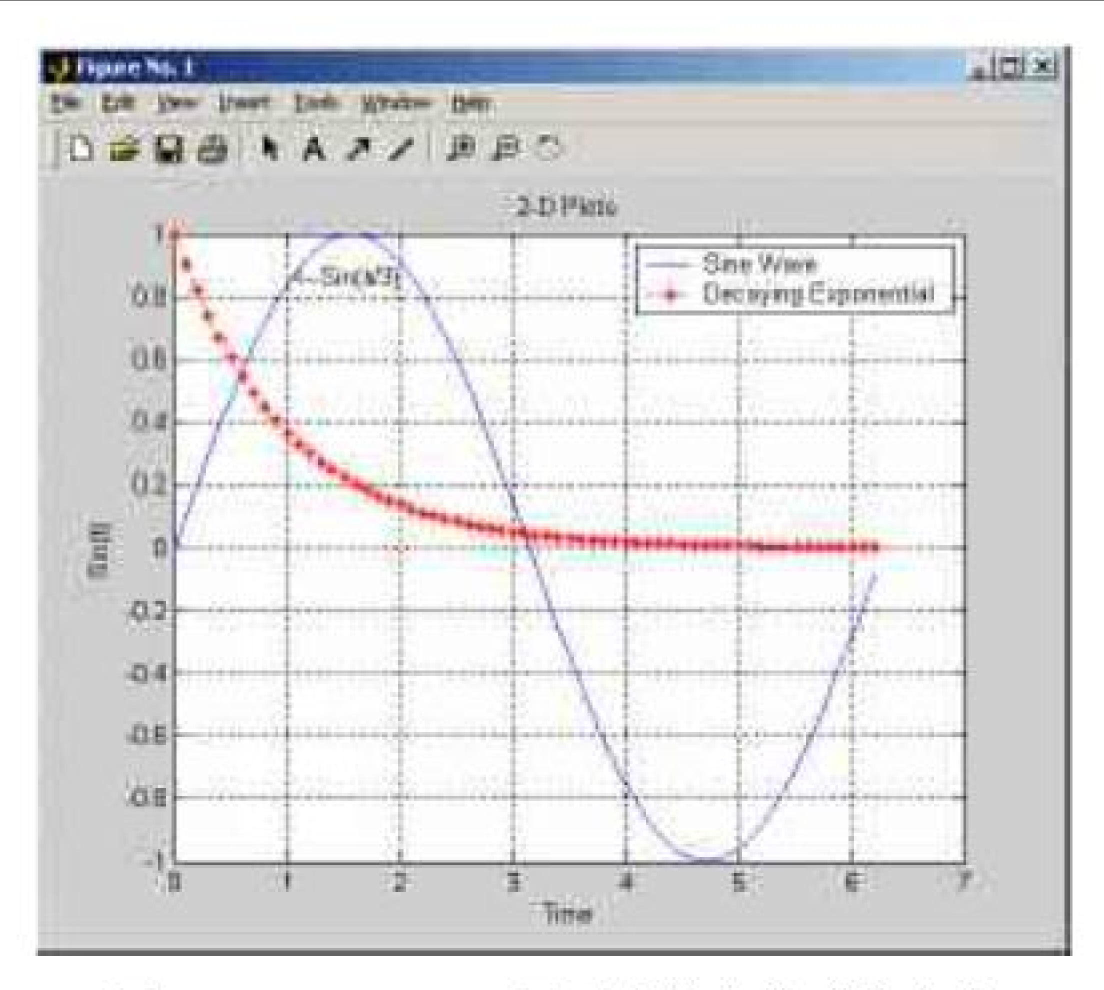


图 6-19　MATLAB 图形窗体的格式化文本

为图形窗体增加这些文本信息一般有多种途径，或者通过菜单命令，或者通过属性编辑器，或者使用 MATLAB 提供的函数。MATLAB 的图形窗体 Insert 菜单下包含多个菜单命令可以用来添加这些格式化的文本信息，而通过 MATLAB 图形编辑器，配合不同对象的属性编辑器也可以完成添加格式化文本信息的工作。不过这些方法都没有利用函数编写程序简便、灵活，所以在本书中，重点介绍利用函数添加格式化文本信息的方法。

1．添加标题(title)

添加图形的标题需要使用 title 函数，该函数的基本用法为

```
title('string')
```

其中，字符串 string 为图形窗体的标题，该标题将被自动地设置在轴的正中顶部，例如在 MATLAB 命令行窗口中，键入下面的指令：

```
>> title(date)
```

则 MATLAB 会创建包含一个空白轴的图形窗体，同时将轴的标题设置为当前的日期，如图 6-20 所示。

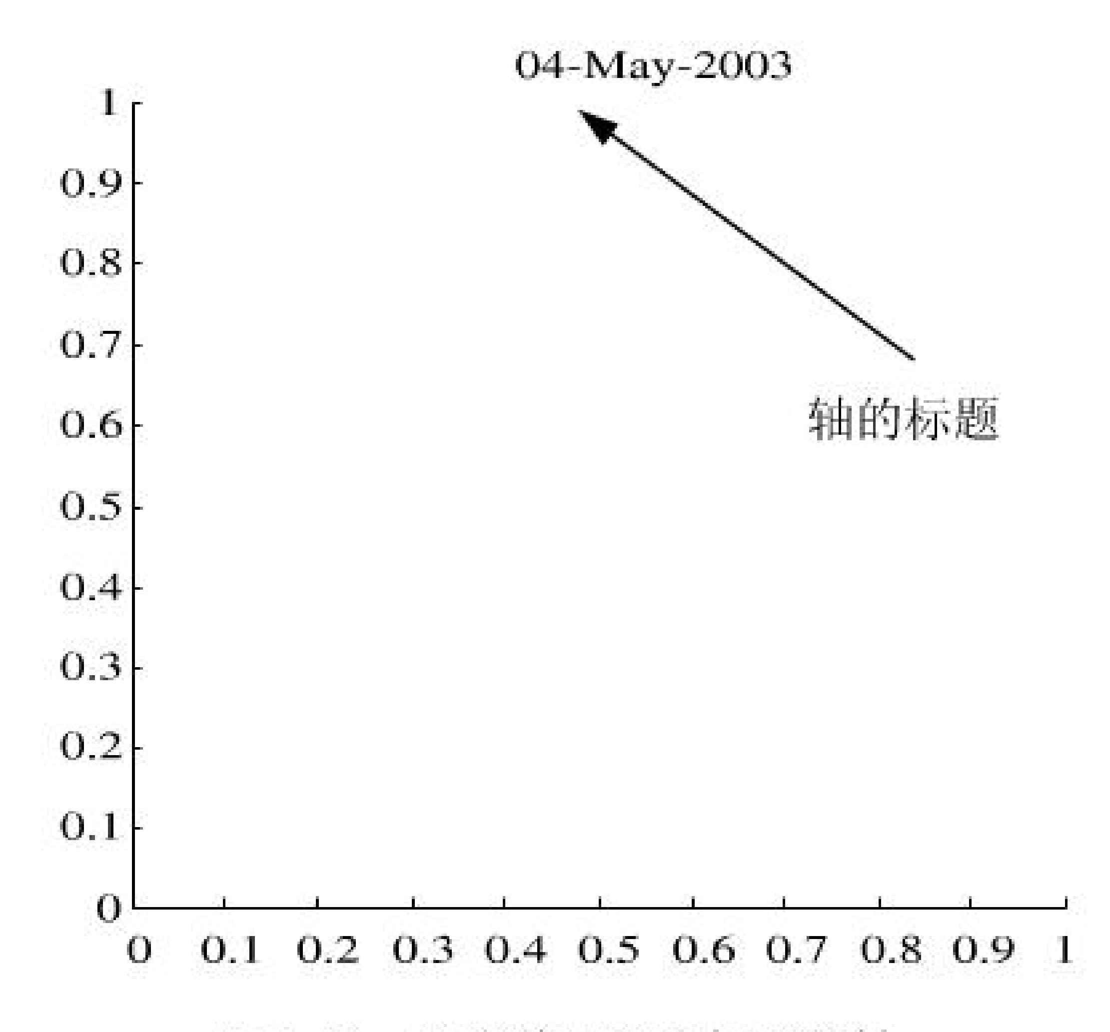


图 6-20　以当前日期为标题的轴

2．添加图例(**legend**)

图例作为绘制在轴的数据曲线的说明，默认绘制在轴的右上角处，其中包括了绘制在轴内曲线的色彩、样式和时标，同时在绘制图例的地方为每一个曲线添加简要的说明文字，便于用户了解数据曲线的信息。添加轴的图例需要使用函数 legend，该函数的基本语法为

```
legend('string1', 'string2'……)
```

其中，字符串 string1、string2 为图例的说明性文本，MATLAB 将自动地按照绘制在轴上的曲线的绘制次序选择相应的文本作为图例。例如，假设在图形窗体上绘制如例子 6-3 所示的三条曲线，为这三条曲线增加图例。在 MATLAB 命令行窗口键入下面的指令：

```
>> legend('y=sin(t)','y=sin(t-pi/2)','y=sin(t-pi)')
```

这时的图形窗体将出现添加好的图例，如图 6-21 所示。

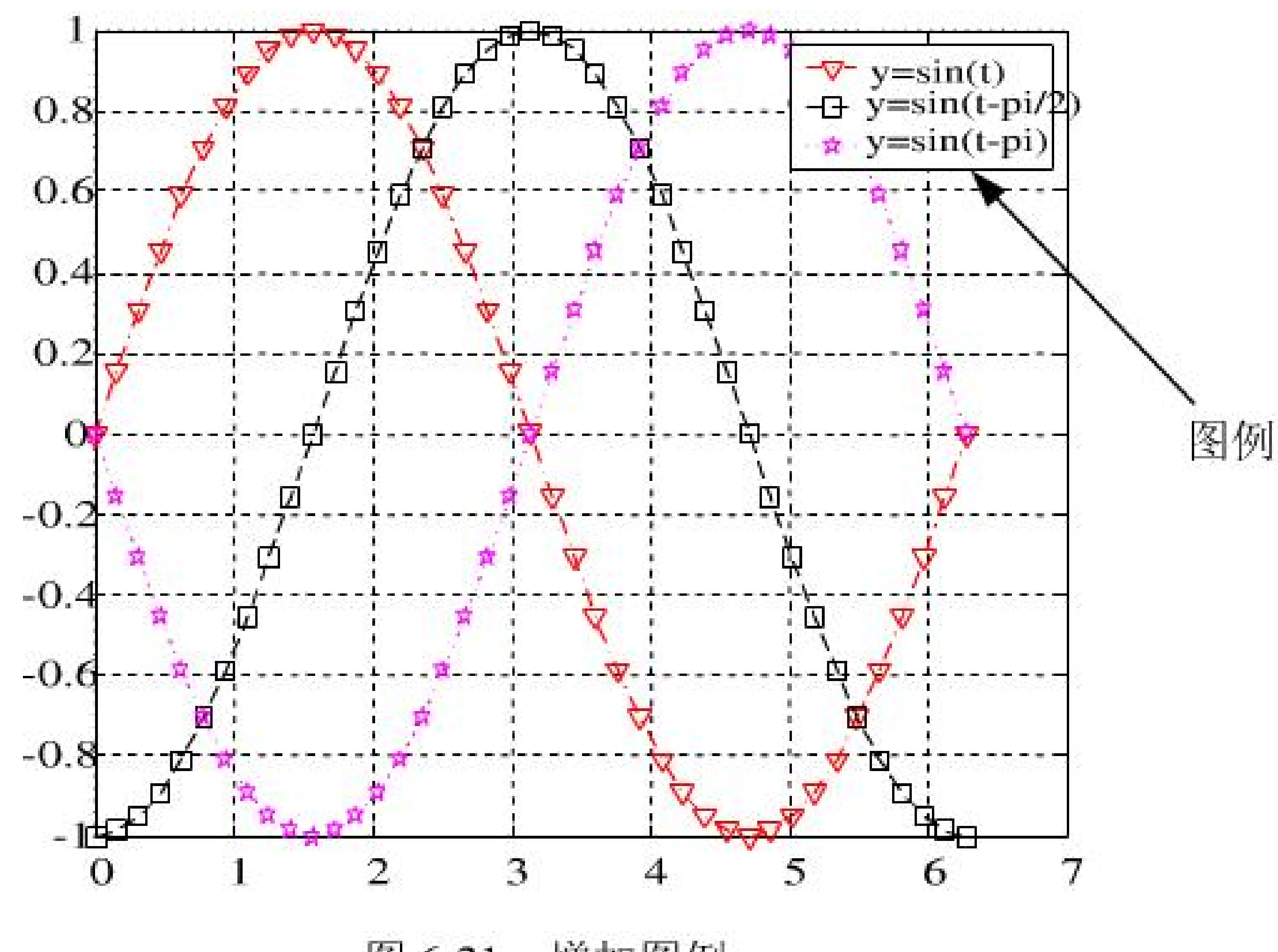


图 6-21　增加图例

通过图例可以非常方便地了解绘制在图形窗体中的曲线的基本信息。图例所在的位置可以任意地挪动，可以用鼠标直接在图形窗体中移动图例的位置，也可以在创建图表的时候，直接利用 legend 函数设置图表的不同位置。具体的方法请参阅 MATLAB 的帮助文档中关于 legend 函数的说明。另外，还可以使用句柄图形的方法设置图例的位置。

3．添加坐标轴标签(**label**)

在 MATLAB 中坐标轴的标签可以用来说明与坐标轴有关的信息，坐标轴标签也可以包含各种需要添加的信息，例如坐标轴数据的单位、物理意义等。MATLAB 可以为不同的坐标轴添加不同的坐标信息，一般地，可以使用 xlabel、ylabel 和 zlabel 函数分别为图形窗体的 X 轴、Y 轴和 Z 轴添加轴标签。以 X 轴为例，这三个函数的基本使用语法如下：

```
xlabel('string')
```

其中，string 就是坐标轴的标签。坐标轴的标签自动与坐标轴居中对齐。

例如在图形窗体中为 X 轴和 Y 轴添加标签：

```
>> plot(sin(0:pi/100:pi))
>> xlabel('X 轴数据');ylabel('Y 轴数据')
```

添加标签之后的图形窗体内容如图 6-22 所示。

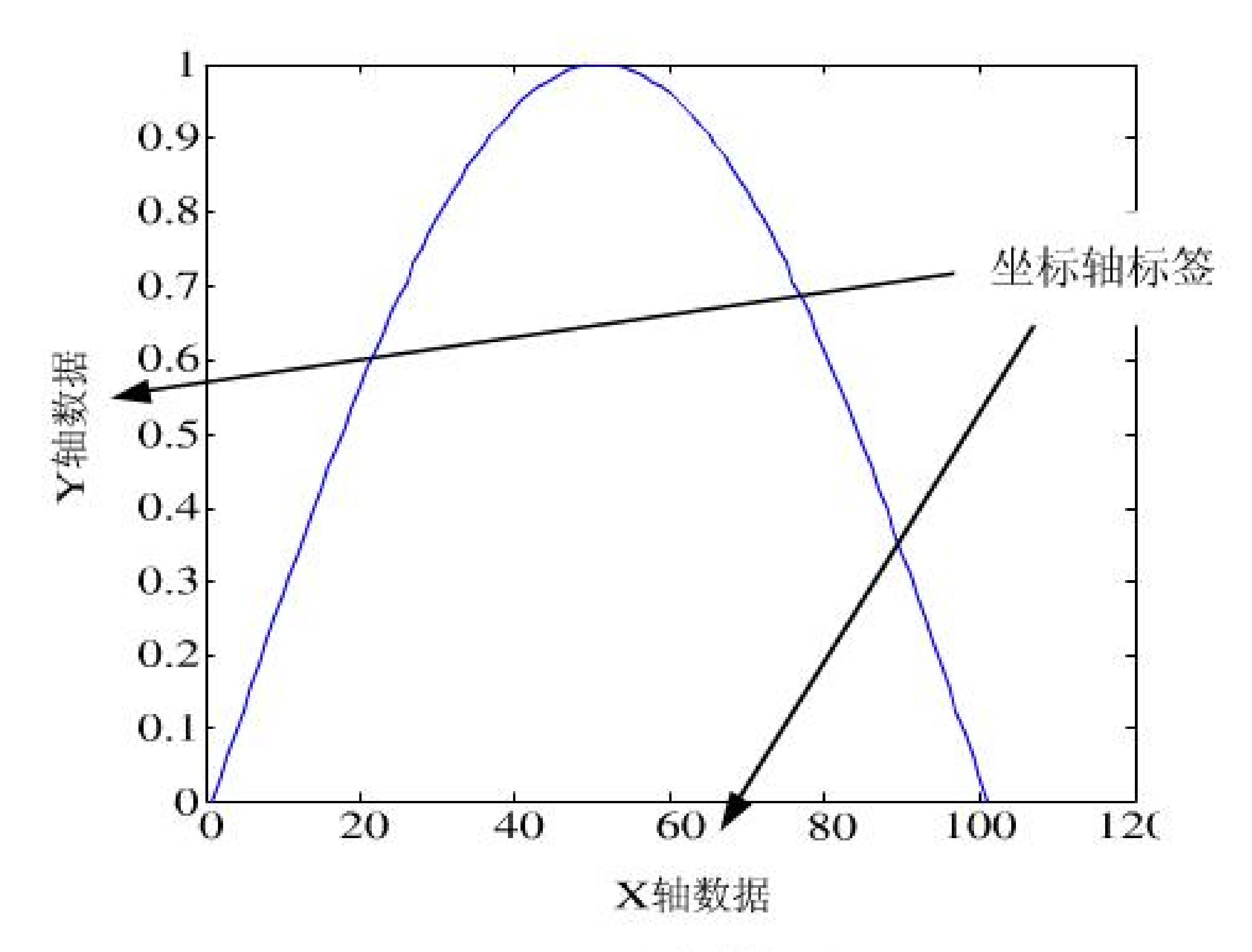


图 6-22　添加坐标轴标签

4．添加文本注释(text)

文本注释是由创建图形的用户添加的说明行文字，这些文字可以用来说明数据曲线的细节特点，比如，需要特别注意的数据点。创建文本注释的时候可以将文本注释首先保存在元胞数组中，然后使用 text 函数完成向图形窗体添加文本注释的工作。

text 函数的基本语法为

text(x, y, 'string')

其中，x 和 y 是文本注释添加的坐标值，该坐标值使用当前轴系的单位设置，这个坐标也就是文本起始点的坐标。

例如可以向图形窗体添加文本：

```
>> x = 0:.1:2*pi;y = sin(x);plot(x,y)
>> text(pi/3,sin(pi/3), '<--Sin(\pi/3)')
```

这里通过调用 text 函数，把文本注释添加到了图形曲线上，如图 6-23 所示。

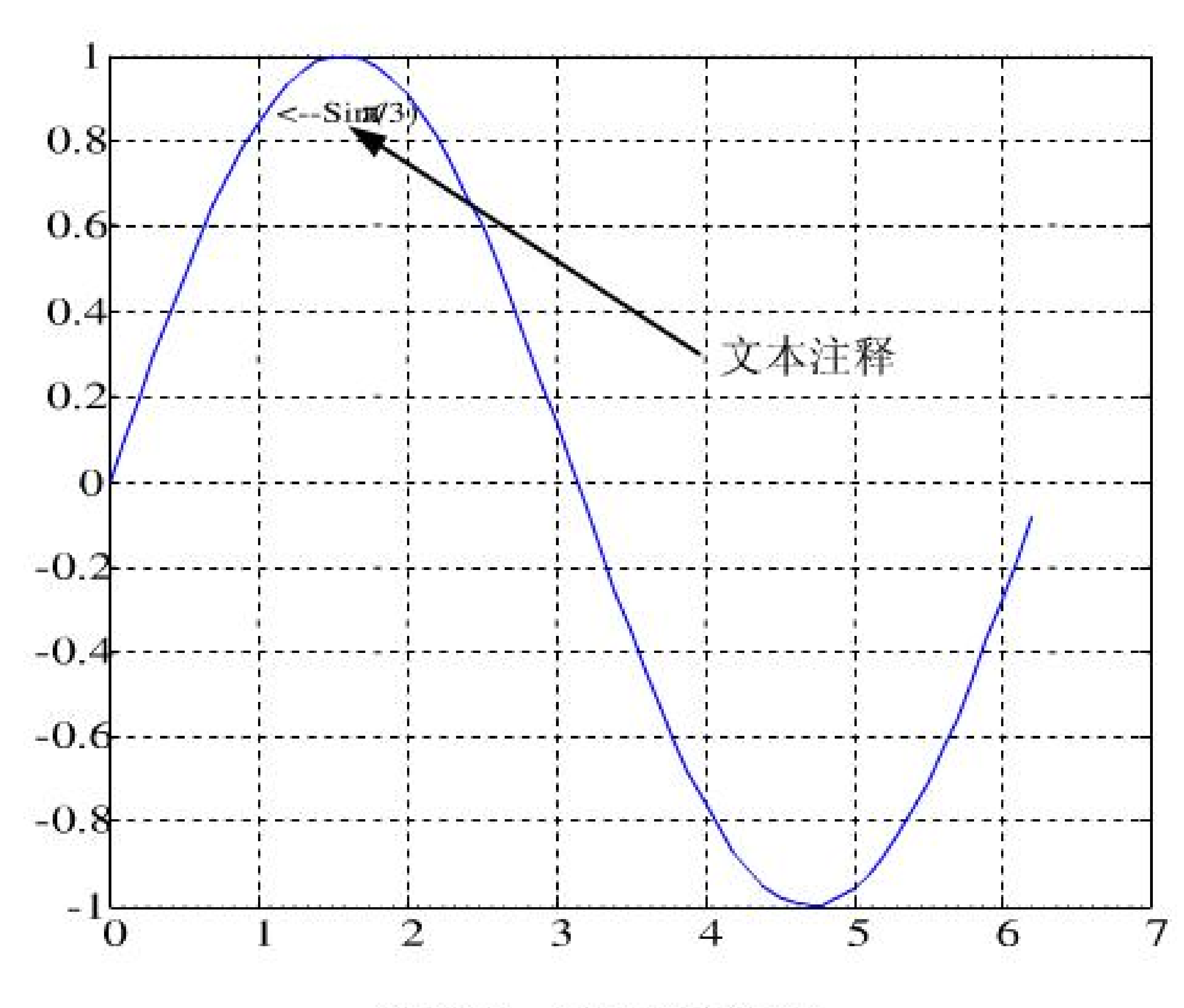


图 6-23　添加文本注释

注意：

在这里使用了 LaTeX 字符来显示常数 p，关于 LaTeX 将在后面的章节详细介绍。

6.3.2　格式化文本标注

在 6.3.1 小节的例子中添加的各种文本标注都使用了系统默认的字体、字号等属性设置，若是所有的图形文本注释都使用这些格式未免显得千篇一律了，所以这些文本的属性也可以进行修改，修改这些属性可以通过 set 命令，而前提是需要获取相应图形对象的句柄。句柄图形已经超出了本书的讨论范围，有兴趣的读者可以参阅 MATLAB 的帮助文档。在本小节介绍创建格式化文本标注的方法。

文本标注的字体属性可以在创建文本标注的时候进行设置，其中有关字体本身的属性包括：

- FontName：字体名称，例如 Courier、隶书等。
- FontSize：字体大小，整数值，默认为 10 points。
- FontWeight：设置字体的加粗属性。
- FontUnits：字体大小的度量单位，默认为 point。

可以在创建文本注释的同时就设置这些文本相应的属性值，在例子 6-9 中，分别设置了文本注释的这些属性。

例子 6-9　添加格式化的文本信息——txtinfo.m。

```
%使用不同的文本标注属性
% 准备数据并绘制曲线
x = 0:.1:2*pi;y = sin(x);plot(x,y)
grid on;hold on
plot(x,exp(-x),'r:*');
% 添加标注
title('2-D Plots','FontName','Arial','FontSize',16)
% 使用中文字体
xlabel('时间','FontName','隶书','FontSize',16)
% 加粗文本
ylabel('Sin(t)','FontWeight','Bold')
% 修改字号
text(pi/3,sin(pi/3), '<--Sin(\pi/3)','FontSize',12)
legend('Sine Wave', 'Decaying Exponential')
hold off
```

执行脚本文件 txtinfo，在 MATLAB 命令行窗口中键入下面的指令：

```
>> clear all
>> txtinfo
```

得到的图形输出如图 6-24 所示。

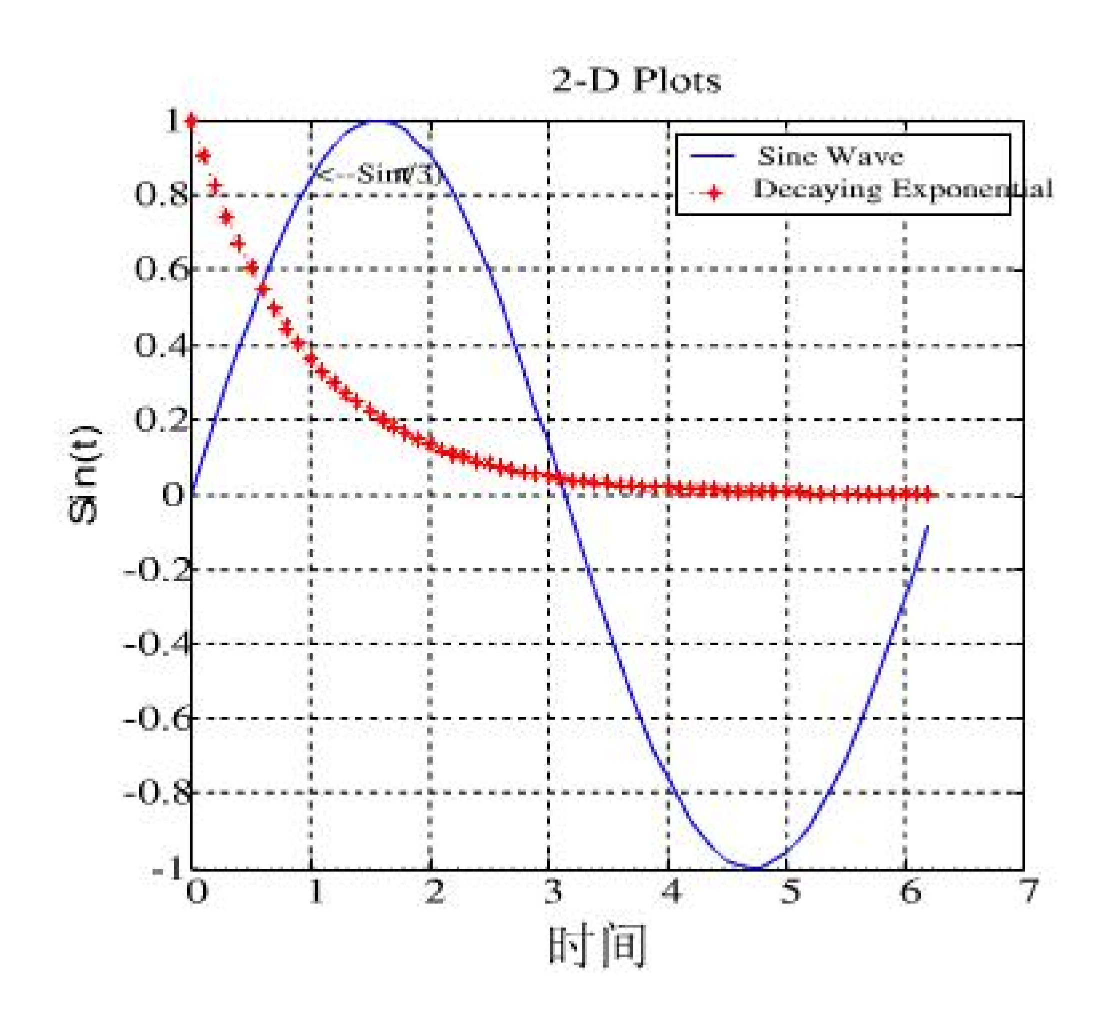


图 6-24　设置不同文本属性的文本标注

注意例子 6-9 的运行结果，由于设置了不同的文本属性，所以图 6-24 中不同的文本具有不同的显示效果。另外，在中文平台上，还具有中文文本字体的支持能力。

若需要修改已经添加好的文本属性，则需要使用句柄图形，或者通过属性编辑器进行修改。

6.3.3 特殊字符标注

在 6.3.1 小节和例子 6-9 中，都使用了特殊的字符集来显示字符 p，而显示字符 p 的方法就是利用了 LaTeX 字符集。利用这个字符集和 MATLAB 文本注释的定义，就可以在 MATLAB 的图形文本标注中使用希腊字符、数学符号或者上标和下标字体等。

提示：

在 MATLAB 可用的 LaTeX 字符集在附录 B 中可以查到。

在 MATLAB 图形窗体的所有文本标注中都可以使用这些特殊的文本，比如在标题、坐标轴标签、文本注释中，使用特殊文本时一定要注意不要忘记"\"符号，否则 MATLAB 就会按照普通文本处理这些字符。除了直接使用附录中的 LaTeX 字符集外，还可以用下面的标识符组合完成更丰富的字体标注。

- \bf：加粗字体。
- \it：斜体字。
- \sl：斜体字(很少使用)。
- \rm：正常字体。
- \fontname{fontname}：定义使用特殊的字体名称。
- \fontsize{fontsize}：定义使用特殊的字体大小，单位为 FontUnits。

设置字体的大小或者名称将直接影响接在定义符后面的文本内容，直到下一个字体定义符出现。

进行上标或者下标文本的注释需要使用"_"和"^"字符。进行上标标注的方法如下：

{superstring}

其中，superstring 是上标的内容，它必须添加在大括号“{}”之中。

进行下标标注时的标注方法如下：

_{substring}

其中，substring 是下标的内容，它必须添加在大括号“{}”之中。

关于在 MATLAB 文本标注中添加特殊的文本的具体方法参见例子 6-10。

例子 6-10　使用特殊文本标注——Latex_examp.m。

```
function latex_examp
%LATEX_EXAMP 在文本注释中使用特殊文本
alpha = -0.5;
beta = 3;
A = 50;
t = 0:.01:10;
y = A*exp(alpha*t).*sin(beta*t);
% 绘制曲线
plot(t,y);
%添加特殊文本注释
title('\fontname{隶书}\fontsize{16}{隶书} \fontname{Impact}{Impact}')
xlabel('^{上标} and _{下标}')
ylabel('Some \bf 粗体\rm and some \it{斜体}')
txt = {'y = {\itAe}^{\alphax}sin(\beta\itt)',...
                    ['\itA\rm' ,' = ',num2str(A)],...
                    ['\alpha = ',num2str(alpha)],...
                    ['\beta = ',num2str(beta)]};
text(2,22,txt );
```

运行例子 6-10，在 MATLAB 命令行窗口中键入指令：

>> latex_examp

得到运行结果如图 6-25 所示。

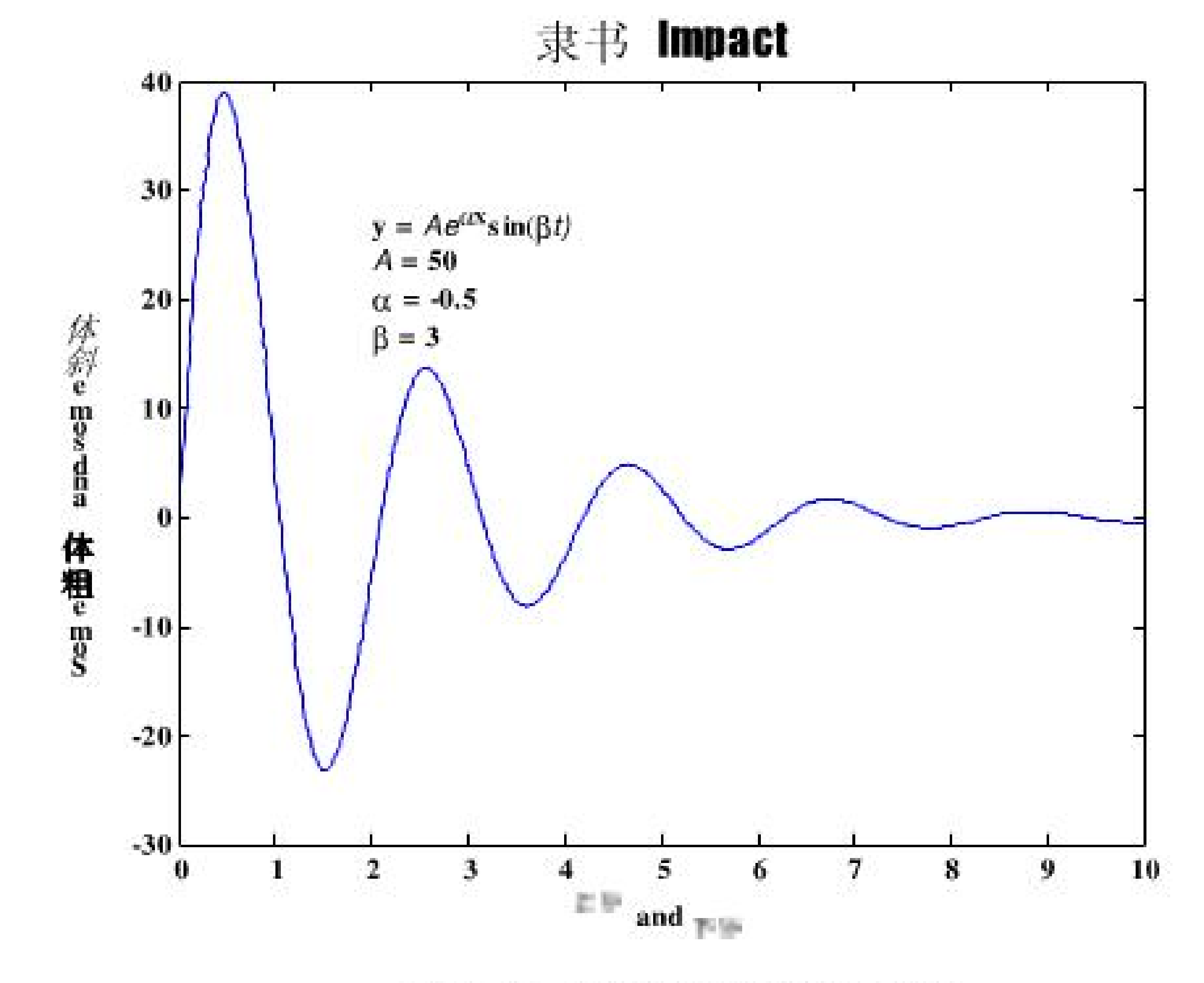


图 6-25　使用特殊文本注释

特殊文本注释可以放置在各种文本注释的内容中，在例子 6-10 中，011~018 行的代码分别在标题、坐标轴标签、文本注释内容中添加了特殊文本。注意，在需要添加多行文本注释的时候，需要将注释的内容保存在元胞数组中，元胞数组的每一个元胞即为注释的一行，就像 014 行创建的变量 txt 一样。

6.3.4 简单数据统计信息

MATLAB 的图形窗体同时提供了简单数据统计的功能，这些功能是通过调用 MATLAB 的基本数学函数完成的。不过在图形窗体中使用数据统计功能，可以将统计的结构直接绘制在 MATLAB 图形窗体中，而且这些结果也能够保存到 MATLAB 的工作空间。本小节通过一个例子来介绍简单数据统计工具的基本用法。

例子 6-11　简单数据统计工具的基本用法。

本例子使用的脚本文件为 census_stats.m，该文件的内容如下：

```
% CENSUS_STATS 简单数据统计工具使用示例
% 加载数据，数据为 MATLAB 自带的 DEMO
load census;
% 绘制曲线
plot(cdate,pop,'ko');
hold on;grid on;
legend('人口');
title('人口普查信息');
xlabel('时间(年)'); ylabel('人口数(百万)');
```

提示：

本例子使用的数据来自于 MATLAB 自带的 Demo，关于该 MATLAB 自带例子的信息请参阅在线帮助：help census。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> census_stats
```

这时得到的图形窗体内容如图 6-26 所示。

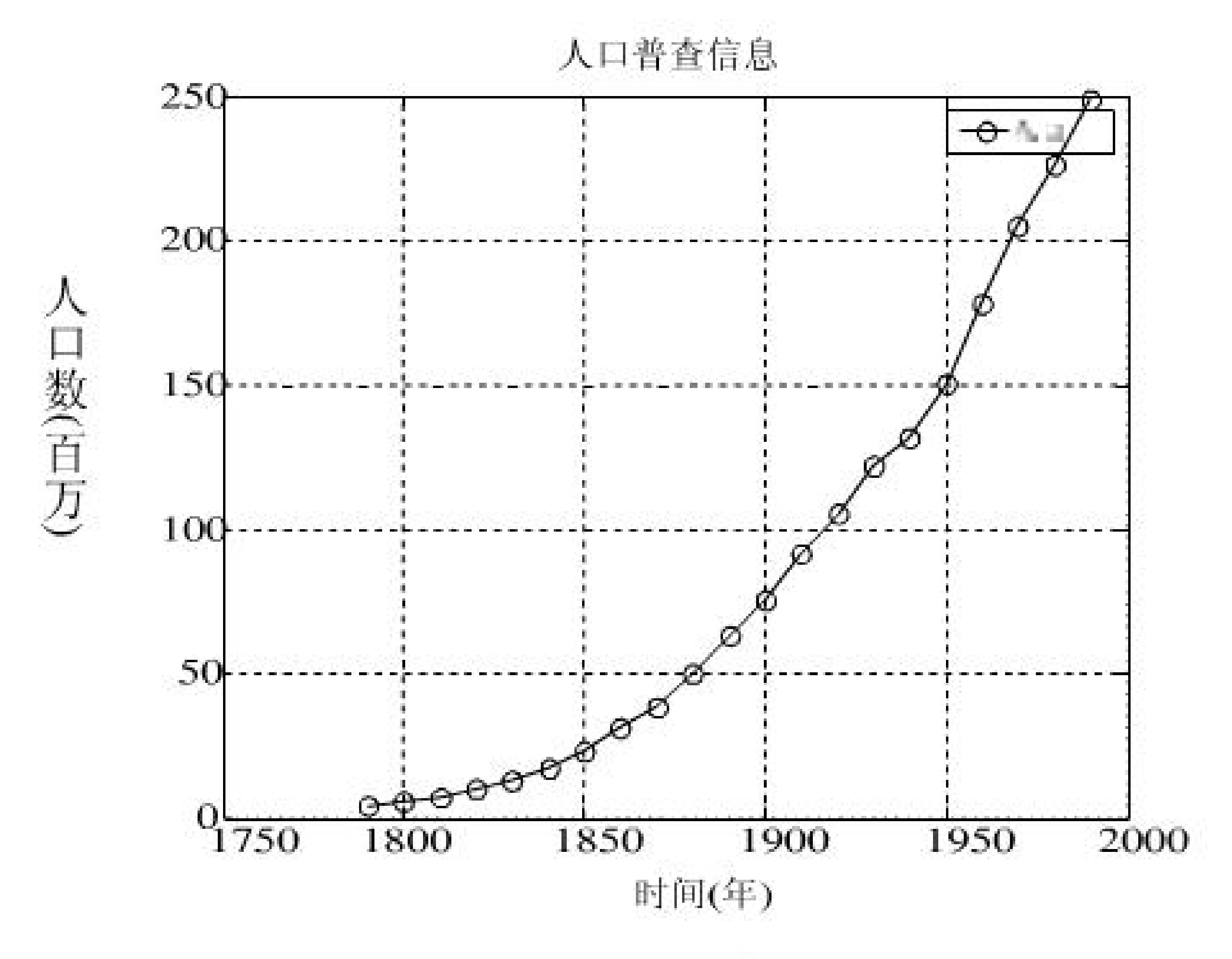


图 6-26　人口数据绘图

为了进行简单数据统计，需要通过菜单命令。执行图形窗体“Tool”菜单下的“Data Statistics”命令，弹出数据统计对话框，如图 6-27 所示。

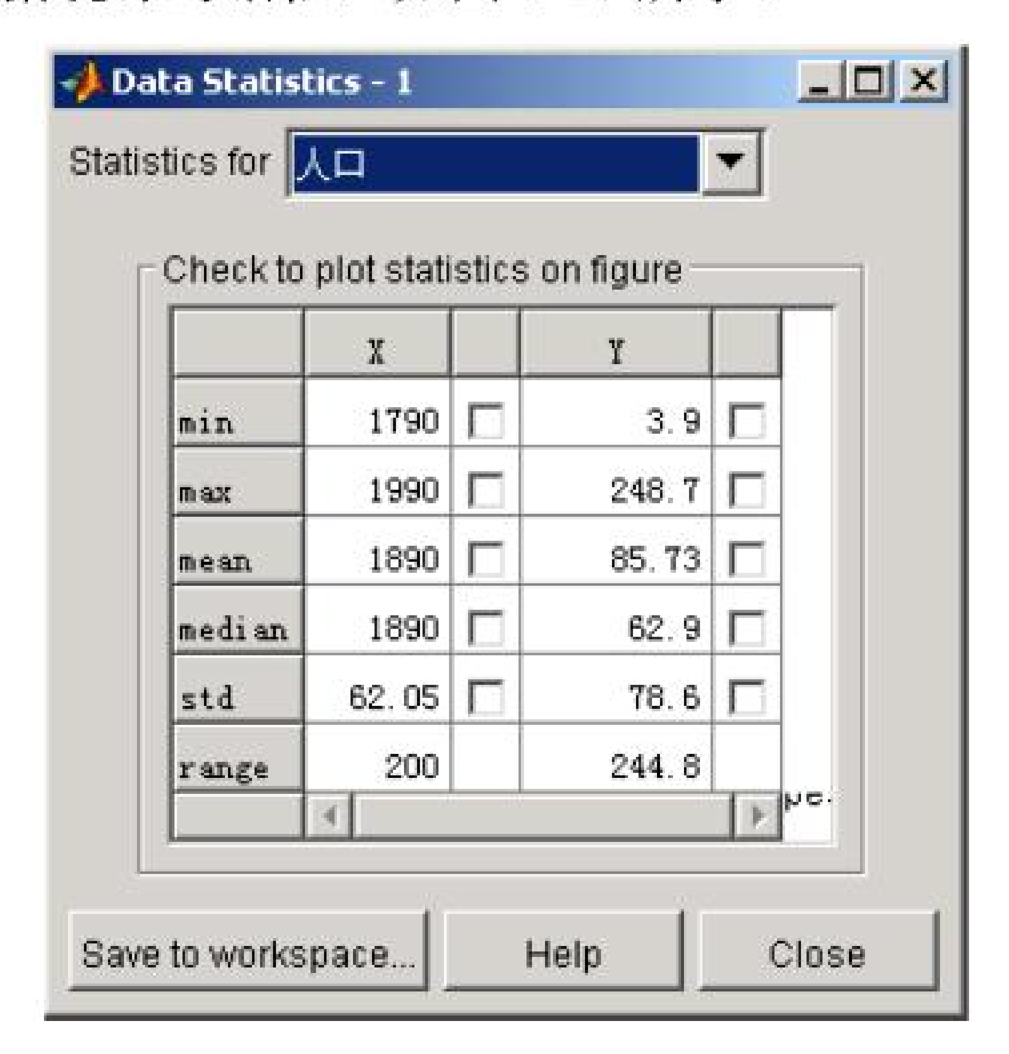


图 6-27　数据统计对话框

在数据统计对话框中，对 X 轴和 Y 轴的数据进行了简要的统计计算，其中包括了最大值、最小值、均值、中值、标准差和取值范围。通过选择每一组数据边上的复选框，就可以将不同的统计计算结果绘制在图形窗体中，比如在例子 6-11 中选择 Y 轴数据的均值(mean)，这时的图形窗体如图 6-28 所示。

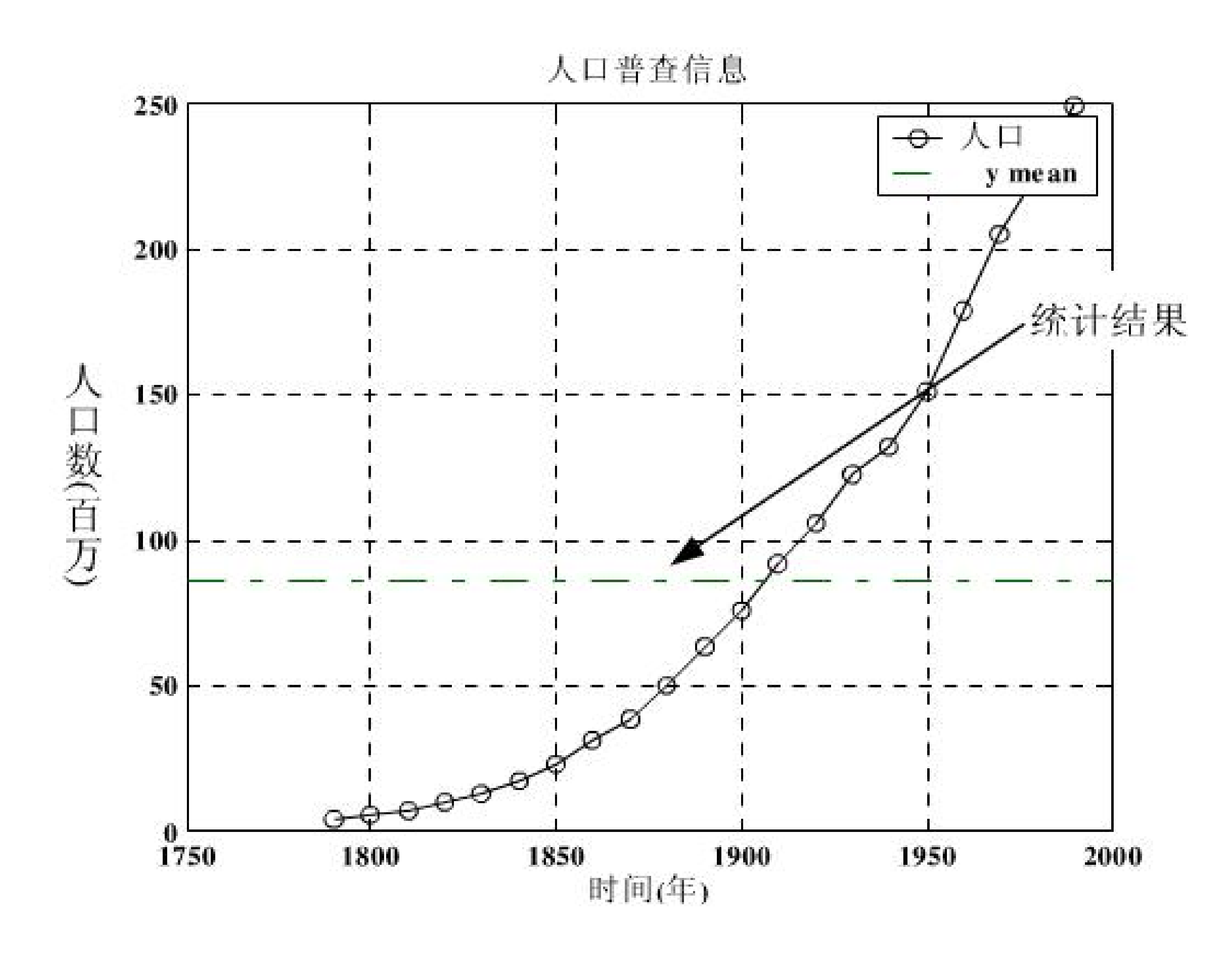


图 6-28　添加数据统计信息

依此类推，可以将其他的统计结果绘制在图形中。

单击数据统计对话框中的“Save to workspace”按钮可以将统计计算的结果保存到工作空间，在弹出的对话框中选择 Y 轴的数据，并且编辑变量名，如图 6-29 所示。

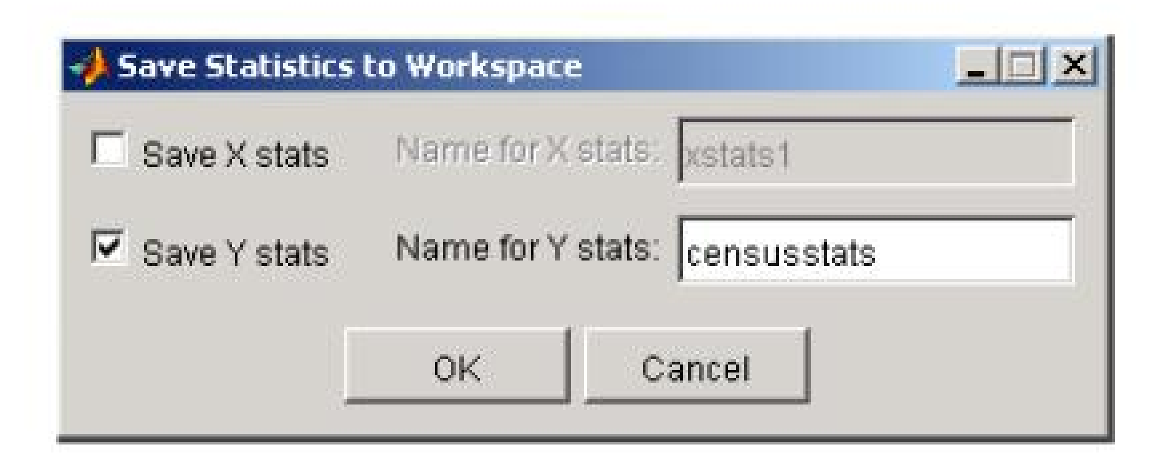


图 6-29　将统计结果保存到工作空间中

保存到工作空间的变量是一个结构：

```
>> whos
  Name            Size              Bytes  Class
  cdate           21x1                168  double array
  censusstats     1x1                 792  struct array
  pop             21x1                168  double array
Grand total is 54 elements using 1128 bytes
>> censusstats
censusstats =
       min: 3.9000
       max: 248.7000
      mean: 85.7286
    median: 62.9000
       std: 78.6011
     range: 244.8000
```

关闭图形窗体的同时自动关闭数据统计工具。若图形窗体中绘制有多条曲线，则可以通过选择数据统计工具对话框的“Statistics for”下拉框中的不同数据进行统计分析。而且，一旦打开数据统计工具对话框，则统计工具自动对图形窗体中的数据进行更新计算，若修改了图形窗体中的曲线，则统计工具自动重新进行计算，并绘制结果。

6.4　特殊图形函数

前面两个小节介绍的都是绘制基本图形曲线的方法，同样对于二维平面绘图也可以使用诸如对数坐标轴等进行表示，另外在 MATLAB 中还能够绘制一些特殊的图形，其中包括柱状图、饼图、火柴杆图等特殊图形。这些特殊的图形主要用于特殊的数据可视化和统计工作中，而且 MATLAB 还能够创建动画，这种动画被称为 MATLAB Movie，除此之外，MATLAB 还能够将图形文件加载到图形窗体中。本小节将介绍实现这些功能的 MATLAB 图形函数以及具体的使用方法。

6.4.1　特殊坐标轴系

MATLAB 为数据的 2D 图形显示提供了很多函数，以下函数都是基本二维画线的函数，

它们采用了不同的坐标刻度——对数坐标系。因此在绘制特殊坐标轴系曲线的时候，需要分别使用不同的函数：

- loglog：两个坐标轴都使用对数刻度。
- semilogx：x 轴用对数刻度，y 轴用普通线性刻度。
- semilogy：y 轴用对数刻度，x 轴用普通线性刻度。

还有一个 plotyy 函数，它能够将两组不相关的数据绘制在同一个图形窗体中，而且每一组数据都使用不同的 Y 坐标系。

例子 6-12　使用特殊的坐标轴系。

执行包含下列代码的脚本文件：

```
%OTHER_AXES 特殊坐标系示例
data = 1:1000;
subplot(2,2,1);loglog(data);grid on;
title('LOGLOG(1:1000)')
subplot(2,2,2);semilogy(data);grid on;
title('SEMILOGY(1:1000)');
subplot(2,2,3);semilogx(data);grid on;
title('SEMILOGX(1:1000)')
subplot(2,2,4);plotyy(data,data,data,data.^2);
grid on;
title('PLOTYY');
```

例子 6-12 执行的结果如图 6-30 所示。

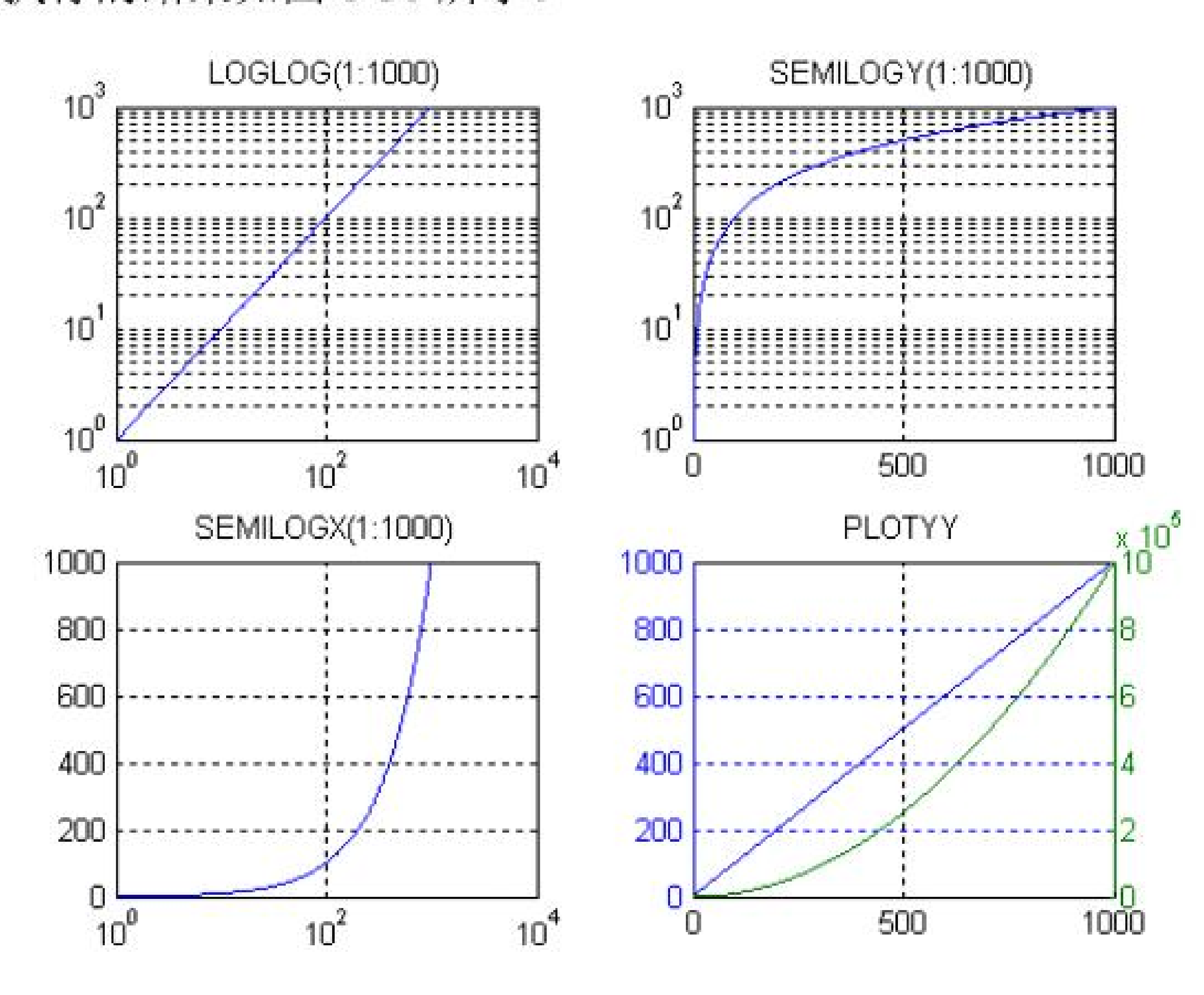


图 6-30　使用特殊的坐标轴系

请读者结合例子 6-12 来检查这些特殊坐标轴系函数的使用方法，也可以参阅 MATLAB 的帮助文档。

6.4.2　绘制特殊图形

在 MATLAB 中能够绘制的特殊图形包括条状图和面积图、饼图、柱状图、离散数据图、矢量方向图以及等高线图等，这些特殊图形的绘制一般都是通过一个函数的调用完成的。不同的特殊图形绘制函数应用面不同，需要根据特殊的数据可视化和统计要求选择。本小节涉及的绘图函数种类众多，在此仅能给出部分函数的用法示例，而无法一一详细解释其语法和用法，请读者注意查阅 MATLAB 的帮助文档或者使用 MATLAB 的在线帮助。

1．条状图和面积图

绘制条状图和面积图的函数如下：

- bar：绘制二维条状图，将 m 行 n 列的矩阵绘制成 m 组，每组 n 个垂直条(bar)。
- barh：绘制二维水平条状图，将 m 行 n 列的矩阵绘制成 m 组，每组 n 个水平条(bar)。
- bar3：绘制三维条状图，将 m 行 n 列的矩阵绘制成 m 组，每组 n 个垂直条(bar)。
- barh3：绘制三维水平条状图，将 m 行 n 列的矩阵绘制成 m 组，每组 n 个水平条(bar)。
- area：绘制面积图，将向量数据绘制成面积图。

执行包含下面代码的脚本文件能够得到如图 6-31 所示的结果。

```
%BAR_EXAMP 条状图和面积图示例
data = [10 2 3 5; 5 8 10 3; 9 7 6 1; 3 5 7 2; 4 7 5 3];
subplot(2,2,[1 2]);bar(data);
title('垂直条状图(2D)');
subplot(2,2,3);bar3h(data);
title('水平条状图(3D)');
subplot(2,2,4);area(data);
title('面积图');
```

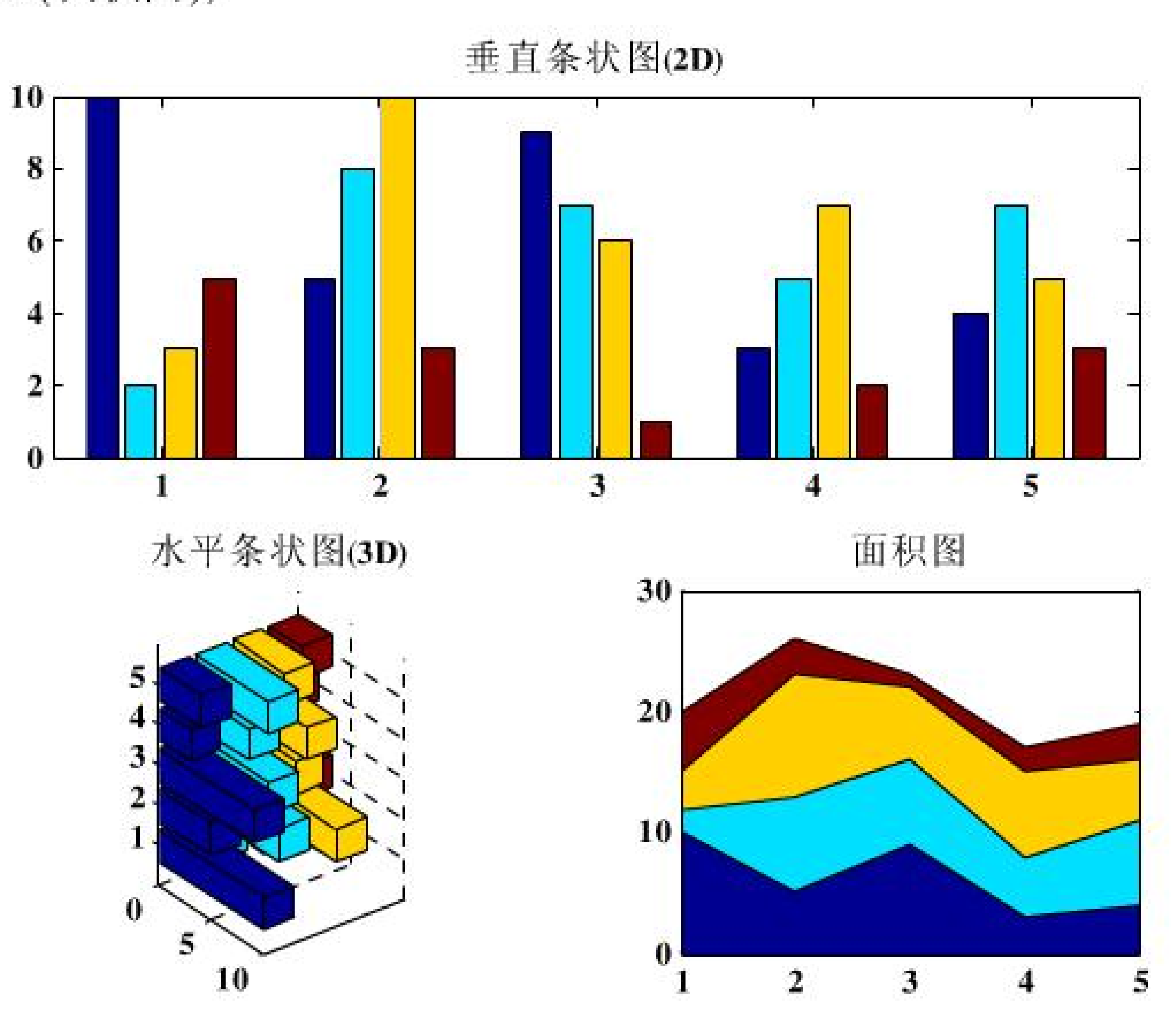


图 6-31　条状图和面积图示例

注意：

条状图处理矩阵依照行元素进行，而面积图处理矩阵则按照列元素进行。

2．饼图

饼图用来显示向量或者矩阵元素占所有元素和的百分比。饼图也有二维饼图和三维饼图，绘制的函数分别为 pie 和 pie3。

执行包含下面代码的脚本文件能够得到如图 6-32 所示的结果。

```
%PIE_EXAMP 饼图示例
A = sum(rand(5,5));
subplot(2,2,1);pie(A);
title('完整饼图(2D)');
subplot(2,2,2);pie3(A);
title('完整饼图(3D)');
B = [0.18 0.22 0.35];
subplot(2,2,3),pie(B);
title('缺角饼图(2D)');
subplot(2,2,4),pie3(B);
title('缺角饼图(3D)');
```

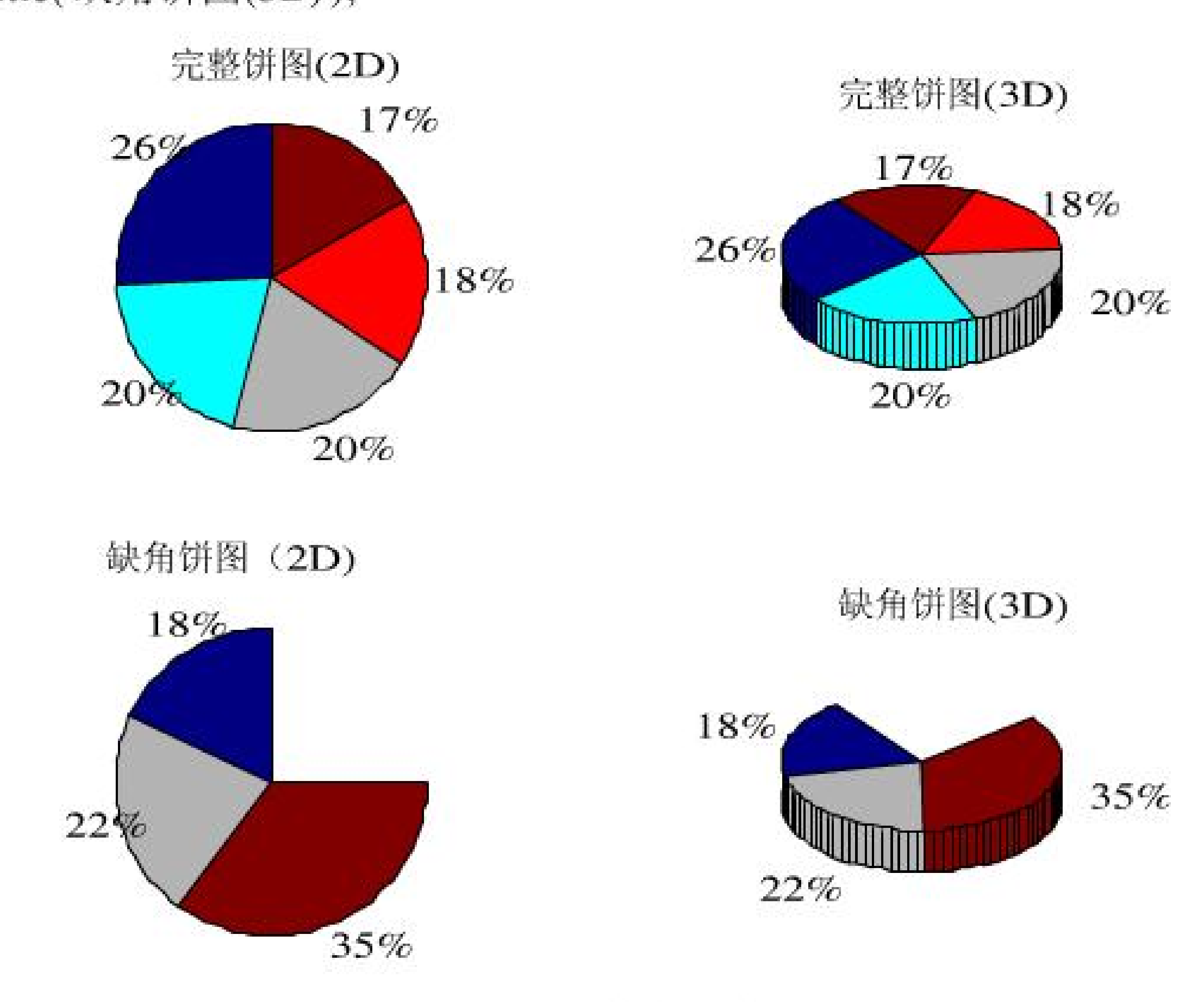


图 6-32 饼图示例

3．柱状图

柱状图(直方图)用来显示数据的分布情况，比如显示一组数据的概率分布情况。柱状图可以绘制在普通的直角坐标下，也可以绘制在极坐标下，使用的函数分别为 hist 和 rose。这两个函数分别计算输入向量中数据落入某一范围的数量，而绘制的柱状高度或者长度则表示落入该范围的数据的个数。

执行包含下列代码的脚本文件能够得到如图 6-33 所示的结果。

```
%HIST_EXAMP 柱状图示例
```

```
A = randn(100000,1);
B = rand(100000,1);
subplot(2,2,1);hist(A);
title('正态分布');
subplot(2,2,2);hist(B);
title('均匀分布');
subplot(2,2,3);rose(A);
title('正态分布');
subplot(2,2,4);rose(B);
title('均匀分布');
```

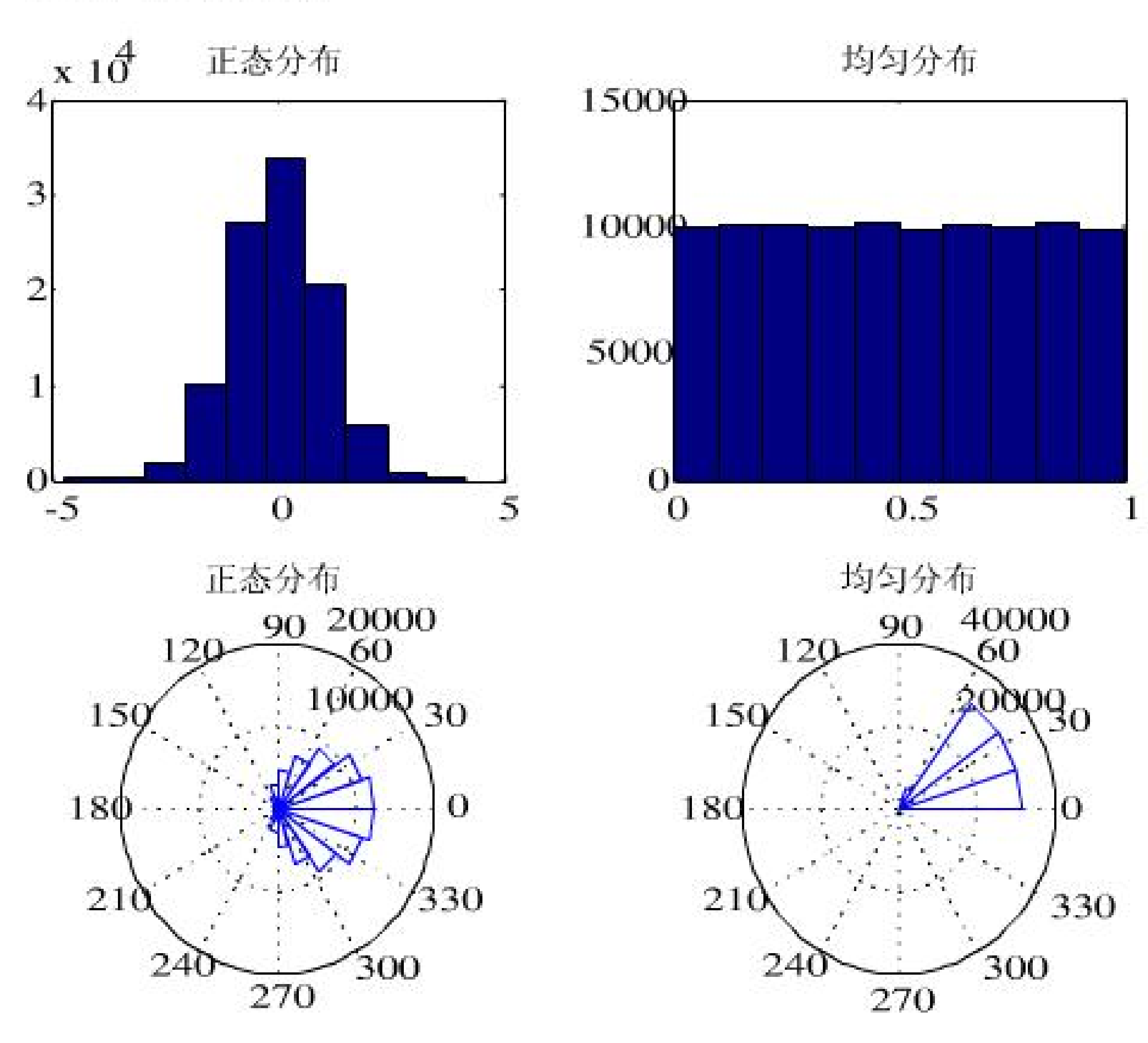


图 6-33　柱状图示例

从柱状图的示例中可以看出两种不同的随机函数的运行结果不同。

4．离散数据图

在数字信号处理领域经常处理一些离散的数据，而 MATLAB 提供了相应的函数供用户进行离散数据的绘制，例如常用的火柴杆图、阶梯图等。前面介绍的柱状图也是绘制离散数据的一种选择。

绘制火柴杆图可以使用 stem 函数或者 stem3 函数，前者绘制二维空间的曲线，后者绘制三维空间的曲线，而阶梯图需要使用 stairs 函数。

例如绘制离散数据图的脚本文件——stem_examp 如下：

```
%STEM_EXAMP 离散数据图示例
alpha = .01; beta = .5; t = 0:0.2:10;
y = exp(-alpha*t).*sin(beta*t);
stem(t,y,'r');grid on;hold on;
stairs(t,y,'g');
```

```
plot(t,y,'b');
figure
theta = 2*pi*(0:127)/128;
x = cos(theta);
y = sin(theta);
z = abs(fft(ones(10,1), 128))';
stem3(x, y, z)
```

上述代码将创建两个数据图，其中一个将二维火柴杆图和阶梯图绘制在一起，另一个为三维的火柴杆图，如图 6-34 所示。

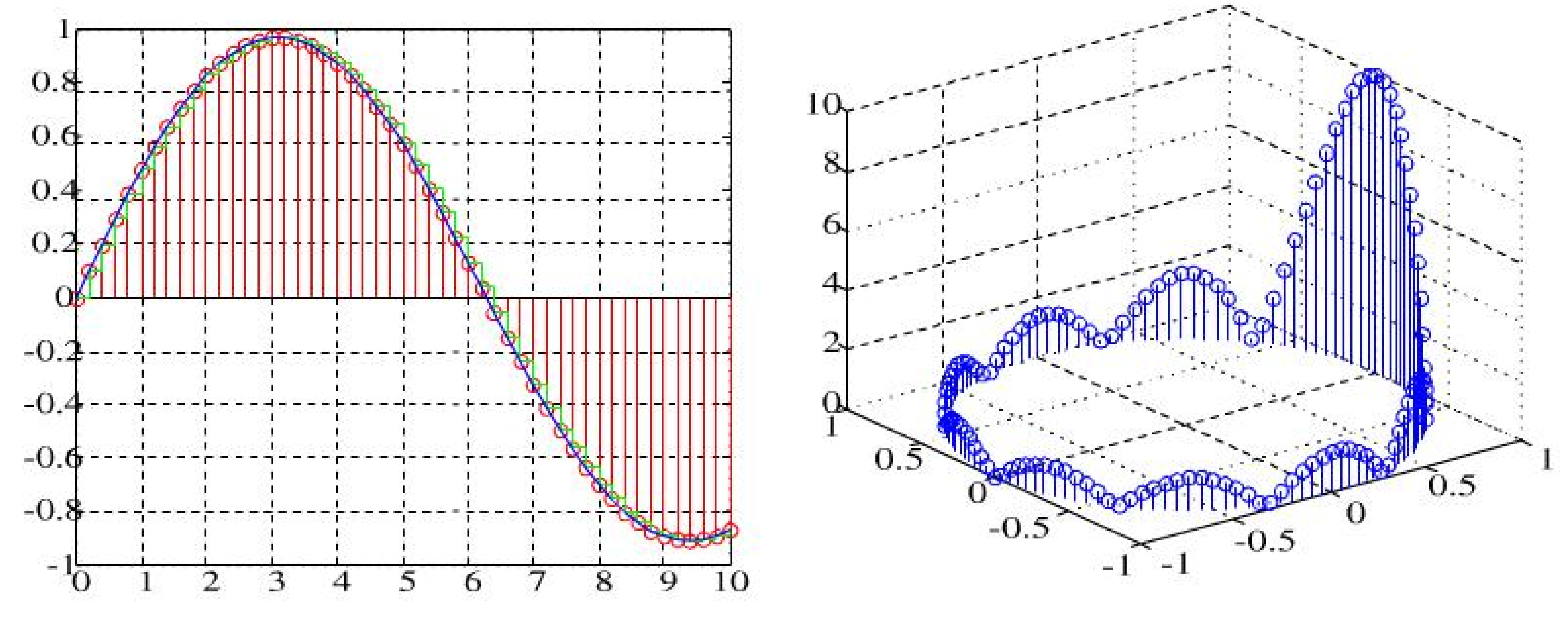


图 6-34 离散数据图示例

注意：

在绘制三维火柴杆图的时候使用了快速傅立叶函数，该函数是数字信号处理常用的函数之一，为 MATLAB 的内建函数。

除了上述四种类型的特殊绘图函数以外，MATLAB 还能够绘制矢量方向图和等高线图，这些函数包括

- compass：绘制放射线图。
- feather：绘制线性放射线图。
- quiver：绘制场图。
- quiver3：绘制三维场图。
- contour：绘制等高线轮廓图。
- contour3：绘制三维等高线轮廓图。
- contourf：绘制填充的等高线图。
- clabel：标识等高线标签。
- meshc：绘制三维 mesh 曲线和等高线。
- surfc：绘制三维 surf 曲线和等高线。

例如使用上述函数绘制特殊图形的脚本文件如下：

```
%OTHERS 矢量方向图绘制示例
subplot(2,2,1)
```

```
[X,Y,Z] = peaks(-2:0.25:2);
[U,V] = gradient(Z, 0.25);
contour(X,Y,Z,10);
hold on
quiver(X,Y,U,V);
title('表面梯度  - (CONTOUR & QUIVER)')
subplot(2,2,2)
contourf(X,Y,Z,10);
title('填充等高线  - (CONTOURF)')
theta = 0:0.1:4*pi;
[x,y] = pol2cart(theta(1:5:end), theta(1:5:end));
subplot(2,2,3)
compass(x,y)
title('放射线图- (COMPASS)')
subplot(2,2,4)
feather(x(1:19),y(1:19))
title('线性放射线图  - (FEAT   ER)')
```

上面的代码运行得到的图形结果如图 6-35 所示。

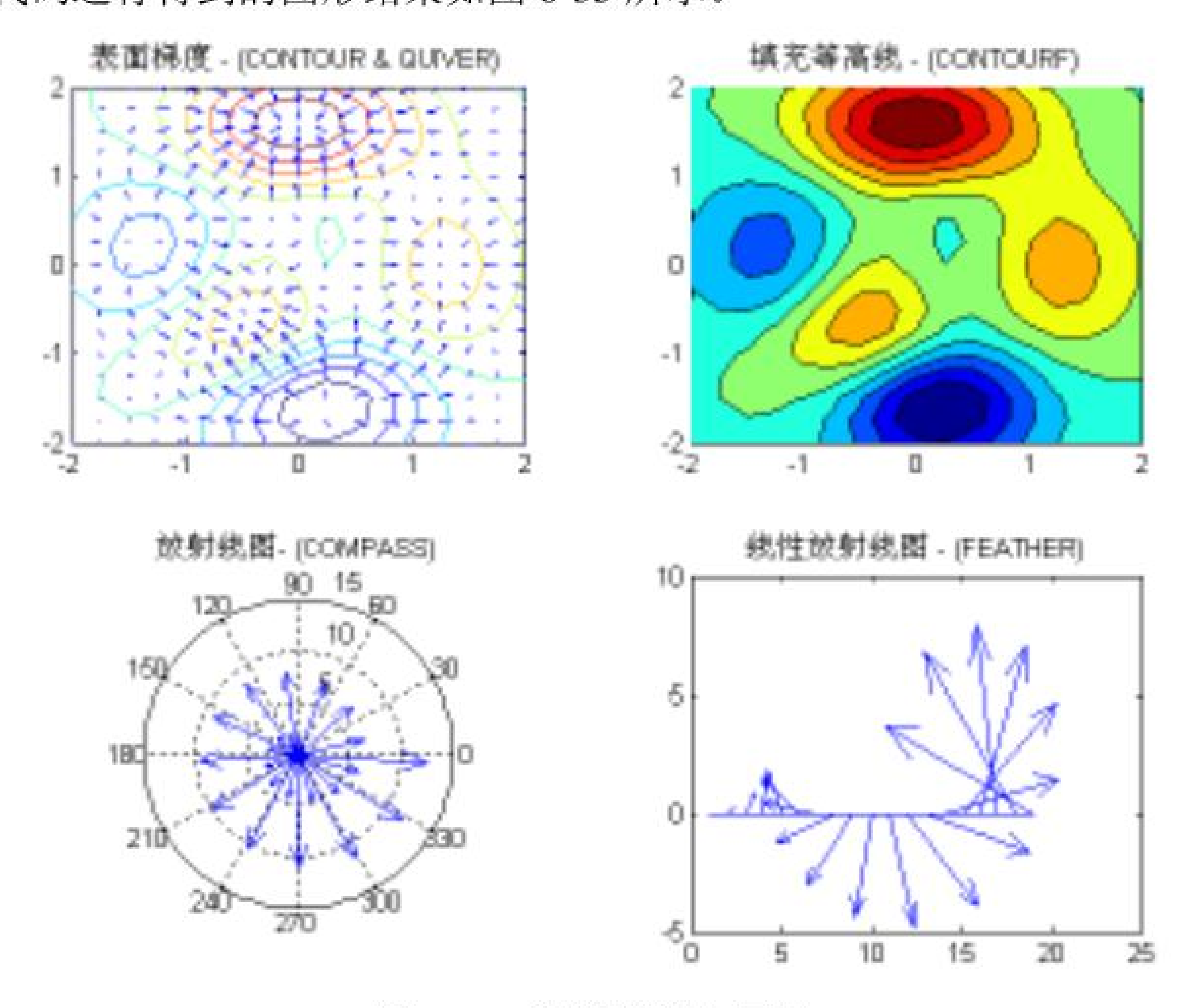


图 6-35　特殊图形绘制示例

注意：

本小节涉及的函数较多，由于篇幅所限没有给出详细的解释，部分函数的用法相对复杂，已经超出了本书的讲述范围。另外三维曲面绘图函数将在本书的 6.5 小节详细介绍。对这些内容有兴趣的读者请参阅 MATLAB 的帮助文档。

6.4.3 调色板(colormap)

MATLAB 为每一个图形窗口都创建了一个调色板(Colormap)。调色板其实就是一个简单的三列矩阵，矩阵的行数就是其定义的色彩数量，而矩阵的每一行三个元素代表着一种色彩，行元素取值范围在 0.0～1.0 之间，这三个元素分别对应一种 RGB 色彩。

MATLAB 默认使用的调色板包含有 64 种色彩，在 MATLAB 命令行窗口中键入 colormap 指令可以获取默认使用的调色板矩阵。

若需要使用调色板矩阵，则需要使用 image 函数绘制数据，然后再使用不同的调色板。其中 image 函数在第五章中就已经用到了，该函数将 MATLAB 工作空间中的矩阵数据绘制在 MATLAB 的图形窗体上。第五章中绘制的数据都是导入的图片，若使用 image 函数绘制普通矩阵会是什么样的结果呢？

例子 6-13　image 函数绘制普通矩阵。

例子 6-13 使用的脚本文件为

```
%IMAGE_EXAMP 使用 image 函数绘制普通矩阵
A = magic(4);
%使用默认的调色板
image(A);
%创建新的调色板
map = hsv(16);
%应用调色板
colormap(map);
%绘制调色板的内容
colorbar;
title('使用 16 色调色板');
```

运行该脚本得到的结果如图 6-36 所示。

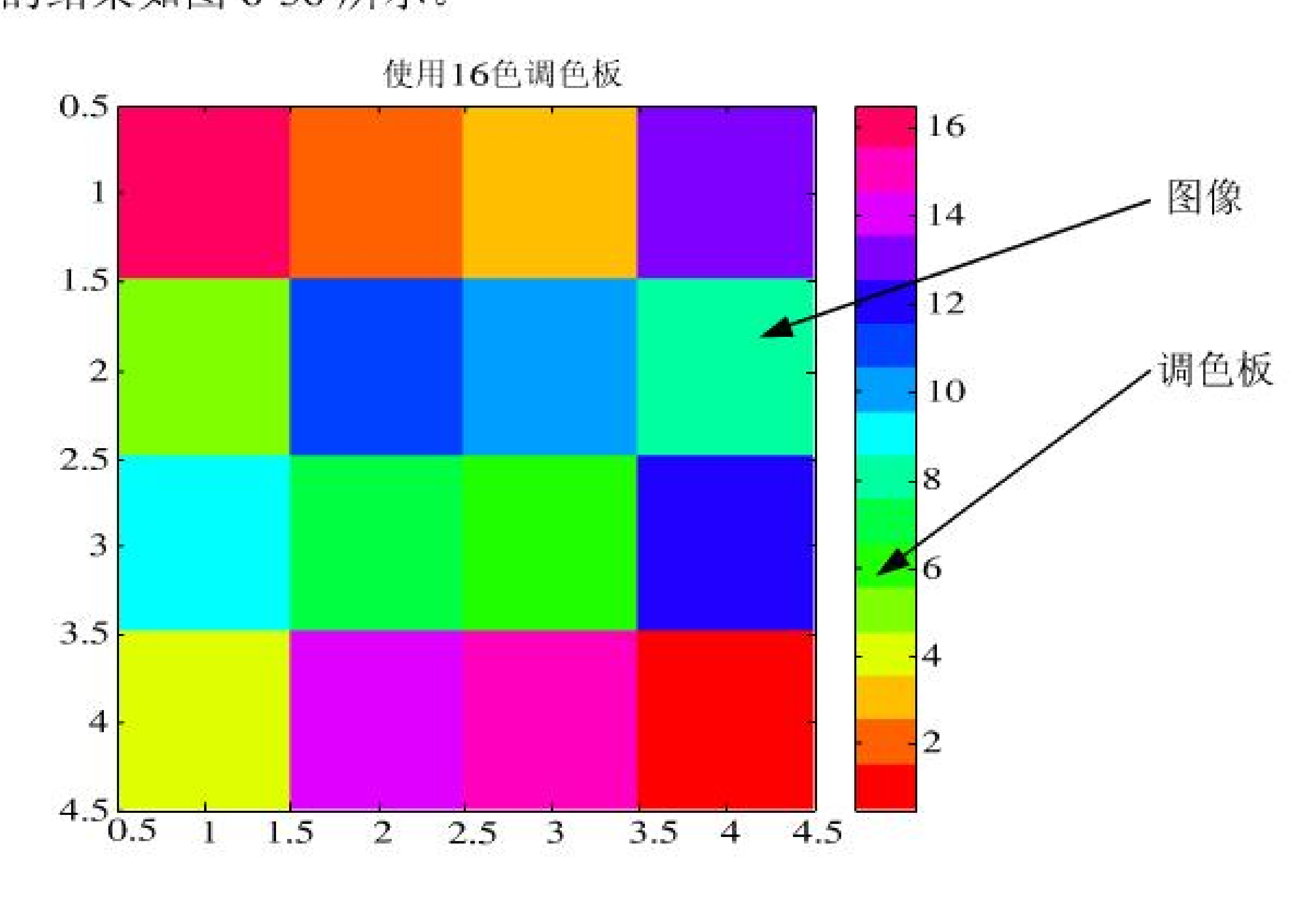


图 6-36　image 函数绘制普通矩阵

尽管例子 6-13 的代码非常短小，还是需要花费一点时间仔细研究一下的。

首先，脚本文件创建了一个具有 16 个元素的幻方矩阵，这个矩阵元素的数值从 1～16 不等。在使用 image 函数将矩阵转变为图像时使用了默认的调色板(有兴趣的读者可以查看一下例子 6-13 的中间图像结果)。

然后脚本文件从 MATLAB 自带的调色板中获取了一个子集，这个子集使用 hsv 函数，将系统提供的 hsv 调色板的前 16 个色彩数据取出，复制给新的调色板——map。接着 colormap 函数使用新创建的调色板，并将调色板的内容绘制在图像右侧，这时才得到例子 6-13 的最终结果。

提示：

colorbar 函数默认将调色板的色彩绘制成垂直的色条，也可以将其绘制为水平的色条，只需指定参数 horiz 即可。

MATLAB 提供了部分默认的调色板，这些调色板分别代表了一种色调，用户可以将这些调色板使用在绘图之中。另外，在图形图像文件格式中，有一种是以索引色保存起来的文件，此类文件加载到 MATLAB 工作空间之后都会有一个调色板矩阵。而应用调色板和绘制色条的方法非常简单，只要在 colormap 函数中指定相应的调色板，然后直接调用 colorbar 函数即可。

例如，在 MATLAB 命令行窗口中键入如下指令可以将 MATLAB 的 logo.bmp 图片文件保存成为索引色位图图片。

```
>> img = importdata('logo.bmp')
img =
          cdata: [192x256 uint8]
       colormap: [255x3 double]
>> image(img.cdata);
>> colormap(img.colormap)
>> colorbar('horiz')
```

这时图片显示的效果如图 6-37 所示。

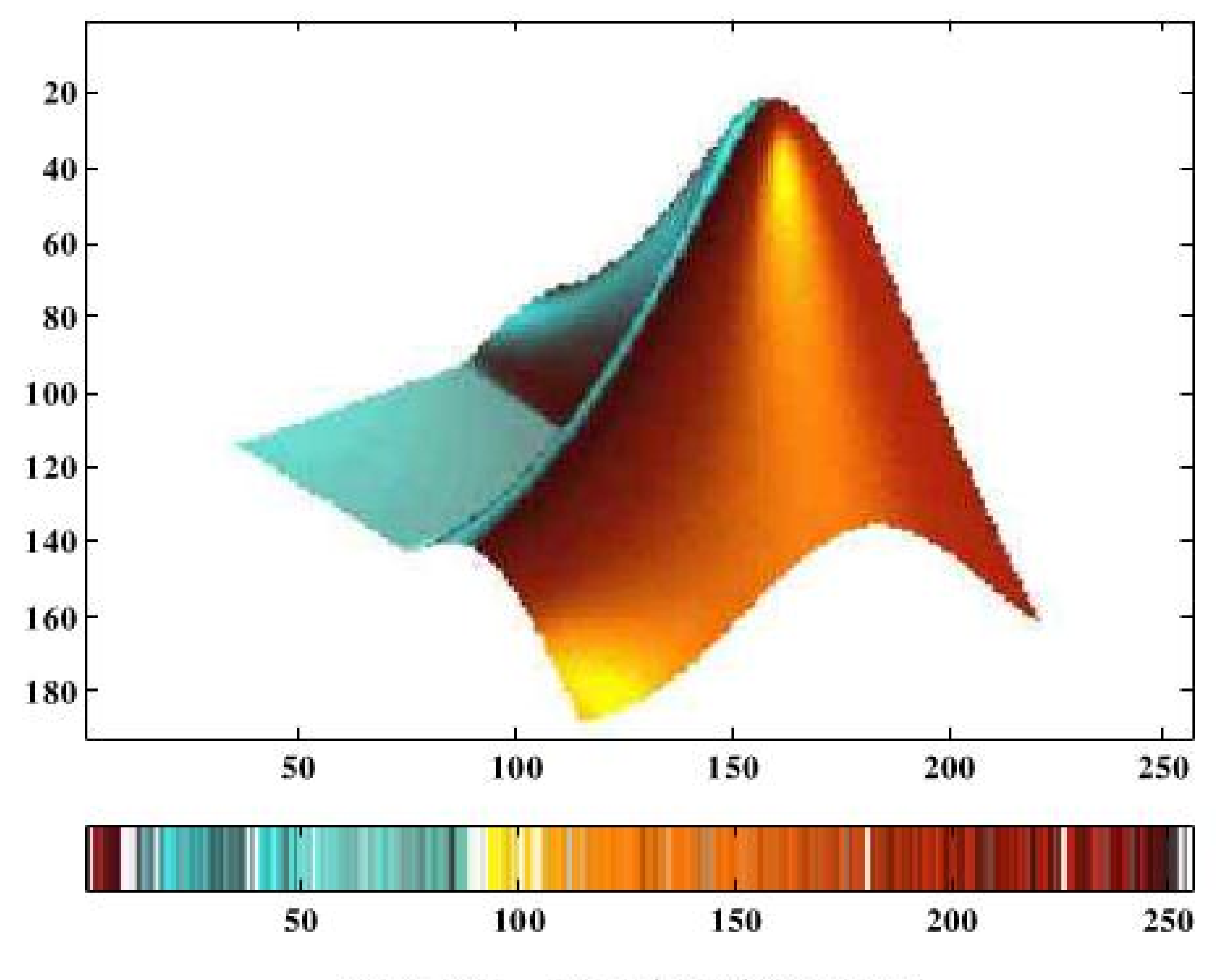


图 6-37　显示索引色的图片

在表 6-2 中，对 MATLAB 提供的标准调色板进行了总结。

表 6-2　MATLAB 的标准调色板

调 色 板	说　明
autumn	红色和黄色为主的色调
colorcube	增强的色彩表，当前系统支持的 RGB 色彩表
cool	青色和洋红色为主的色调
copper	线性的黄铜色调
flag	以红、白、蓝、黑四种色彩为主的色调
gray	线性的灰阶色调
hot	以黑、红、黄、白四种色彩为过渡色的色调
hsv	HSV 色彩模型的色调
jet	HSV 色彩模型调色板的另一种形式，系统默认的调色板
lines	由 MATLAB 绘制曲线使用的默认色彩组成的调色板
pink	粉红色为主色的柔和过渡色调
prism	由棱镜分光得到的色彩组成的调色板
spring	洋红色和黄色为主的过渡色色调
summer	绿色和黄色为主的过渡色色调
vga	Windows VGA 16 色
white	白色
winter	蓝色和绿色为主的过渡色色调

在表 6-2 中总结的这些调色板都是相应的 MATLAB 函数，除了 vga 调色板外，每个调色板都可以指定不同的色彩个数。例如，创建一个具有 256 种色彩的增强色(Colorcube)调色板，命令行如下：

```
map = colorcube(256)
```

除了 MATLAB 系统提供的这十几种标准的调色板以外，MATLAB 还允许用户自己定义调色板。用户自定义的调色板需要通过 MATLAB 提供的调色板编辑器定义。使用该调色板编辑器的方法很简单，在 MATLAB 命令行窗口中键入命令：

```
>> colormapeditor
```

若此时已经存在一个打开的 MATLAB 图形窗体，则调色板编辑器加载当前使用的调色板，否则打开一个空白的图形窗体，然后显示系统默认使用的调色板，如图 6-38 所示。

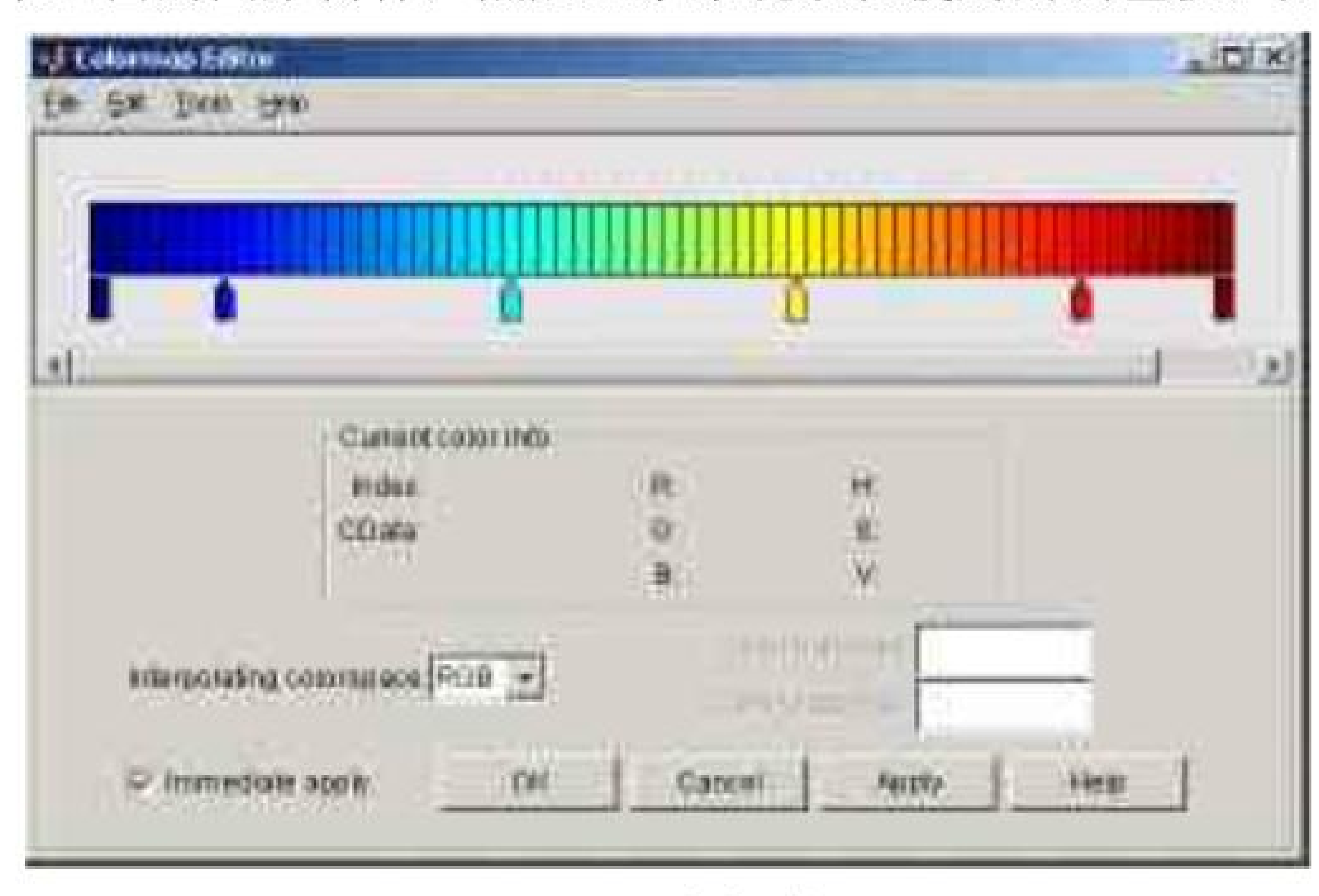


图 6-38　调色板编辑器

用户在编辑自己的调色板时，可以通过加载标准调色板，然后再修改其中的色块的方法来进行。修改色块的时候，只要双击需要修改的色块，系统就会弹出色彩选择对话框(如图 6-39 所示)，供用户选择不同的色彩。若需要删除已有的色块，只要用鼠标选择准备删除的色块，然后按 Delete 键就可以完成操作。若需要添加色块，只要在色彩条的下方的空白处单击鼠标，就可以向已有的调色板中添加新的控制点色块。控制点色块可以任意地在色条上移动，从而得到最后需要的效果。



图 6-39　色彩选择对话框

提示：

在 MATLAB 的帮助文档中有一个例子详细地解释了使用 MATLAB 调色板编辑器的方法，在本书中就不再赘述了，有兴趣的读者可以通过阅读该例子详细了解调色板编辑器的使用方法。

查找该例子的方法为：在 MATLAB 命令行窗口中键入 doc colormapeditor，在函数的帮助中就包含有关例子的说明。

6.5　基本三维绘图

三维图形的表现能力要强于二维图形，在很多时候需要使用 MATLAB 绘制三维图形的能力。MATLAB 提供了若干函数进行三维数据可视化，同时还有若干种方法进行三维图形对象属性的设置和控制。在本小节，将介绍 MATLAB 进行三维图形处理的基本方法。其实三维图形的绘制也是在二维平面中实现的，这其中涉及了计算机图形学等学科的基本知识，有兴趣的读者可以参阅有关的教科书。

绘制三维图形的基本过程要比绘制二维图形复杂一些，基本过程如下：

(1) 准备需要绘制在 MATLAB 图形窗体中的数据。

(2) 创建图形窗体，并且选择绘制数据的区域。

(3) 使用 MATLAB 的 3D 绘图函数绘制图形或者曲线。

(4) 设置调色板和投影算法。

(5) 增加光照，设置材质。

(6) 设置视点(viewpoint)。

(7) 设置绘图坐标轴的属性。

(8) 设置透视比。

(9) 为绘制的图形添加标题、轴标签或者标注文本等。

(10) 打印或者导出图形。

并不是所有三维绘图的过程都包含上面的 10 个步骤，在例子 6-14 中，进行了一次最简单的三维绘图。

例子 6-14　简单三维绘图。

例子 6-14 的脚本文件代码如下：

```
%PLOT_3D 简单三维绘图
% 准备数据
z = 0:0.1:40;
x = cos(z);
y = sin(z);
clf;
% 绘制曲线
plot3(x,y,z)
% 添加标注
grid on
title('Spiral Plot - using PLOT3')
xlabel('x')
ylabel('y')
zlabel('z')
```

该脚本文件运行的结果如图 6-40 所示。

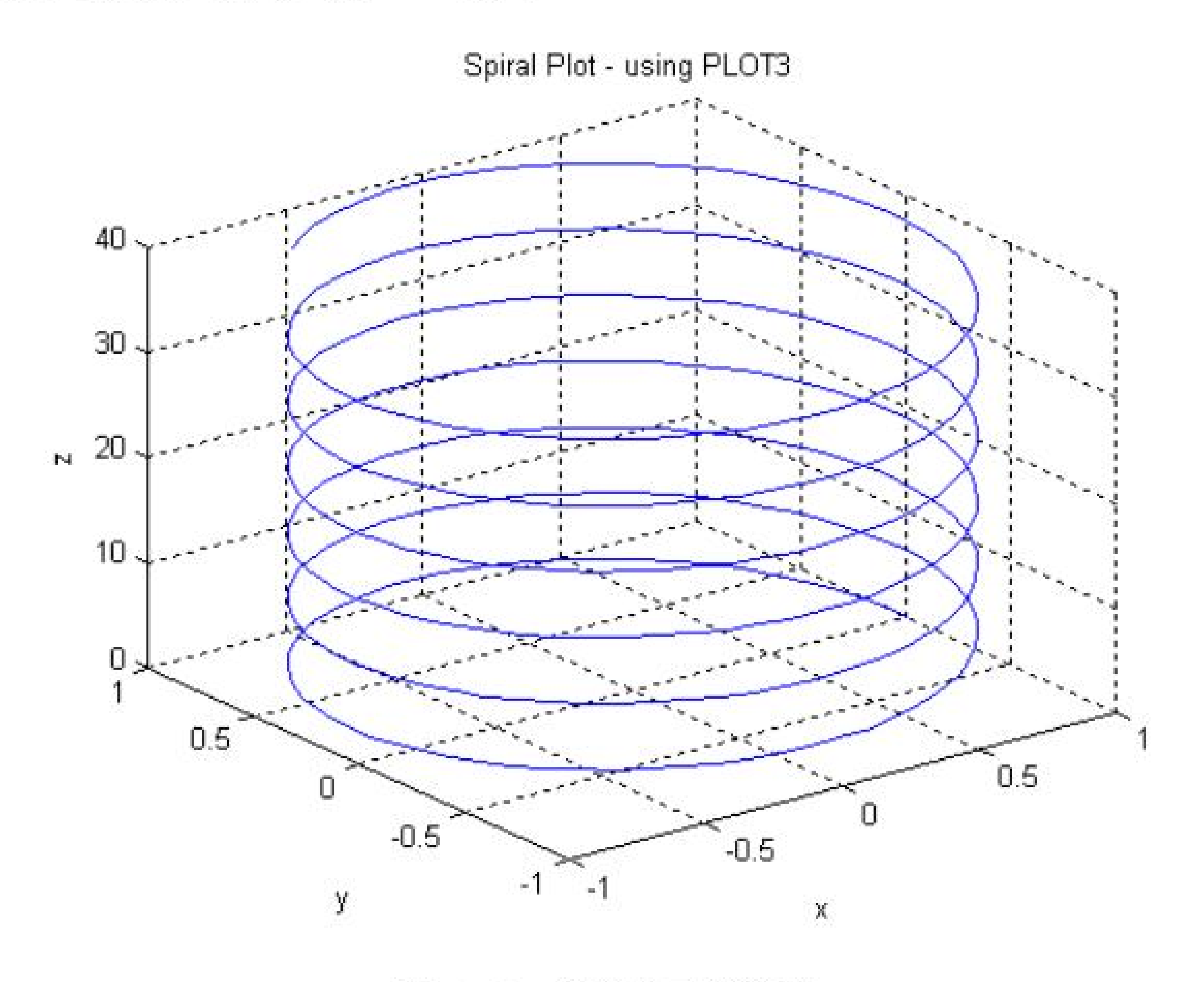


图 6-40　简单的三维绘图

例子 6-14 中使用了函数 plot3，该函数类似 plot 函数，能够将 X、Y、Z 坐标绘制在三维的空间，它的基本语法如下：

plot3(xdata, ydata, zdata, 'clm', ……)

在命令行中，clm 的取值和 plot 函数的取值完全一致。

在例子 6-14 的代码中，使用 zlabel 函数向坐标轴添加标签，它的用法类似于 xlabel 和 ylabel 函数。

在例子 6-14 中，没有进行其他复杂的操作，比如设置光线、视点和三维实体的表面材质等特性。在例子 6-15 中完整地演示了绘制三维曲面的过程。

例子 6-15　绘制复杂的三维曲面。

例子 6-15 使用的脚本文件如下：

```
%准备数据
Z = peaks(20);
%选择图形窗体
figure(1);clf
%调用 3D 绘图函数
h = surf(Z);
%设置调色板和投影算法
colormap hot;
shading interp;
set(h,'EdgeColor','k')
%增加光照
light('Position',[-2,2,20])
lighting phong
%设置材质
material([0.4,0.6,0.5,30])
set(h,'FaceColor',[0 0.7 0.7],...
        'BackFaceLighting','lit')
%设置视点
view([30,25])
set(gca,'CameraViewAngleMode','Manual')
%设置轴属性
axis([0 20 0 20 -8 8])
set(gca,'ZTickLabel','Negative||Positive')
%设置透视比
set(gca,'PlotBoxAspectRatio',[2.5 2.5 1])
%添加文本注释
xlabel('X Axis');ylabel('Y Axis');zlabel('Function Value');
title('Peaks');
```

为了便于观察例子 6-15 的运行结果，可以在调试状态下一步一步地执行代码，例如在执行到代码的 018 行后得到的图形如图 6-41 所示。

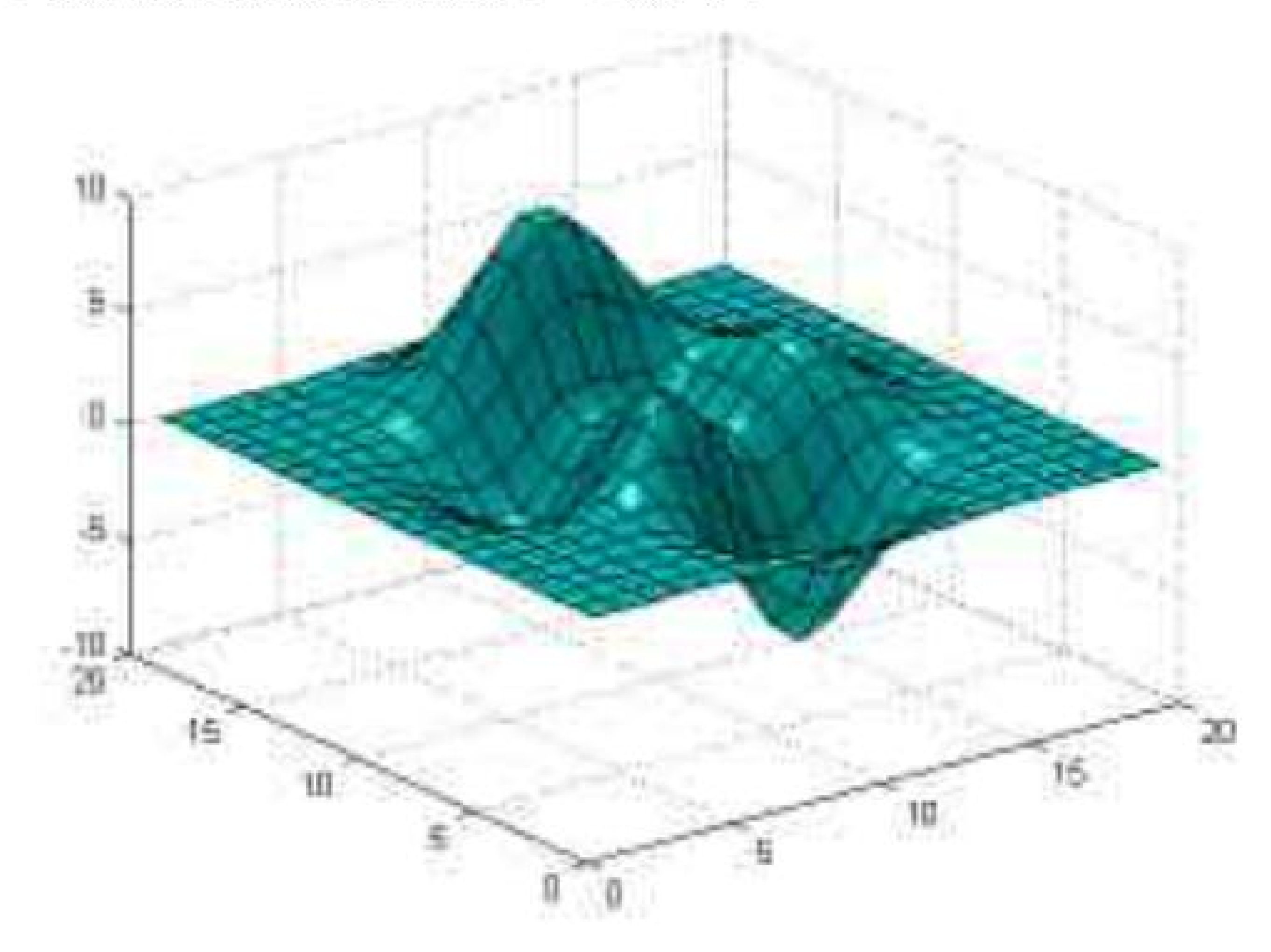


图 6-41　代码运行的中间结果

在运行完毕所有代码后，得到的效果如图 6-42 所示。

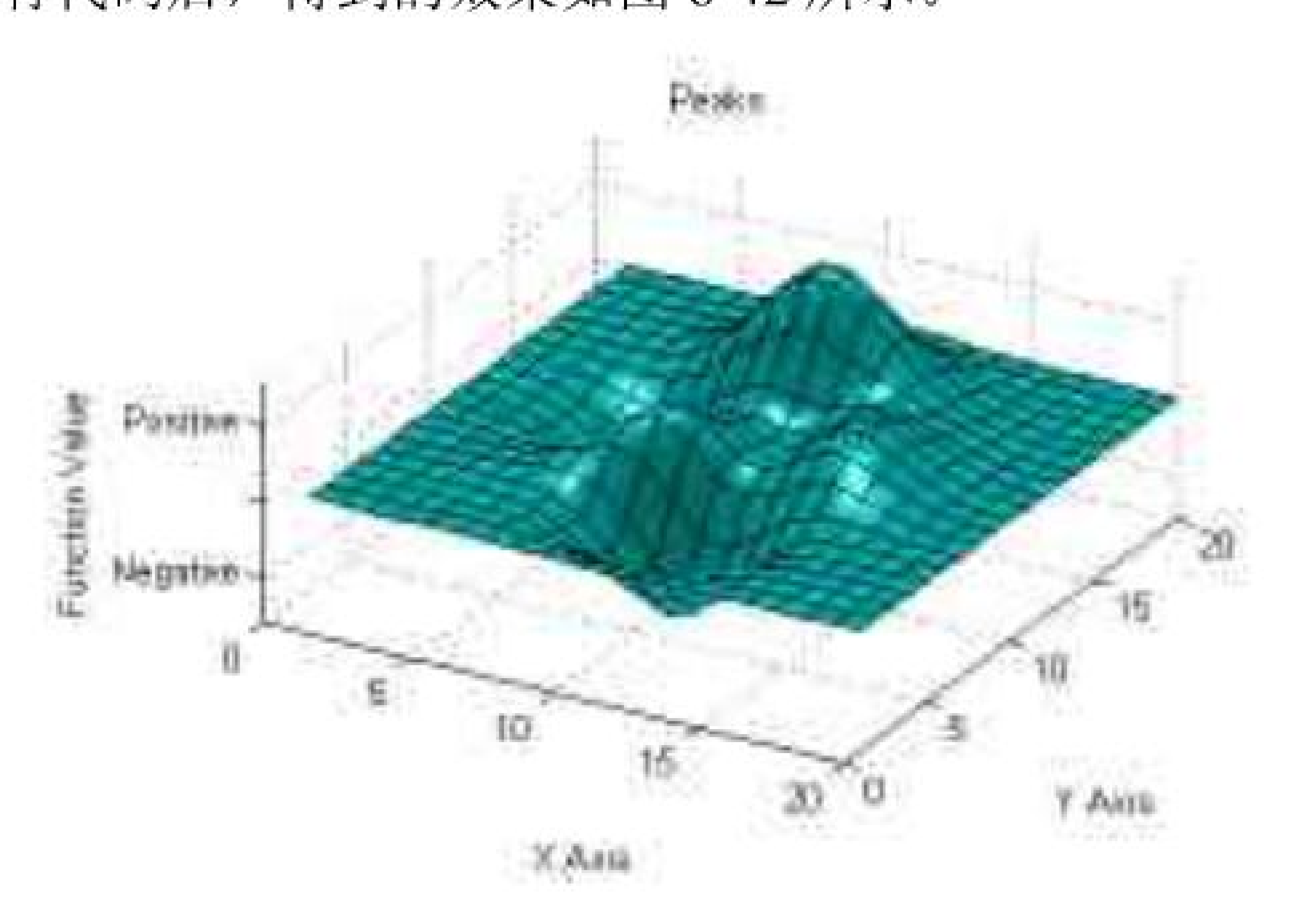


图 6-42　程序运行的最终结果

在例子 6-15 的代码中使用了很多三维图形属性设置的函数以及大量的图形对象属性，这些内容已经超出了本书的讨论范围，有兴趣的读者可以结合例子的代码阅读 MATLAB 的帮助文档或者函数的在线帮助。

为了便于绘制三维图形，MATLAB 提供了一些函数用于特殊的三维曲面绘制，其中经常使用的函数有 mesh 函数和 surf 函数。

mesh 函数用来绘制三维的线框图，它的输入参数一般为 X、Y 和 Z 三个坐标系的数据，同时该函数还有 meshc 和 meshz 函数两种变形，其中 meshc 函数用来绘制具有等高线性质的 mesh 曲面，meshz 函数用于绘制 mesh 曲面的参考面，这三个函数的使用参见例子 6-16。

例子 6-16　mesh 函数的应用。

例子 6-16 的脚本文件包含下列代码：

```
% MESH_EXAMP mesh 函数举例
% 准备数据
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
subplot(1,3,1);
meshc(X,Y,Z);
axis([-3 3 -3 3 -10 5]);title('Meshc');
subplot(1,3,2);
meshz(X,Y,Z);
axis([-3 3 -3 3 -10 5]);title('MeshZ');
subplot(1,3,3);
mesh(X,Y,Z);
axis([-3 3 -3 3 -10 5]);title('Mesh');
colormap gray
    set(gcf,'Position', [14 237 997 275]);
```

例子 6-16 的运行结果如图 6-43 所示。

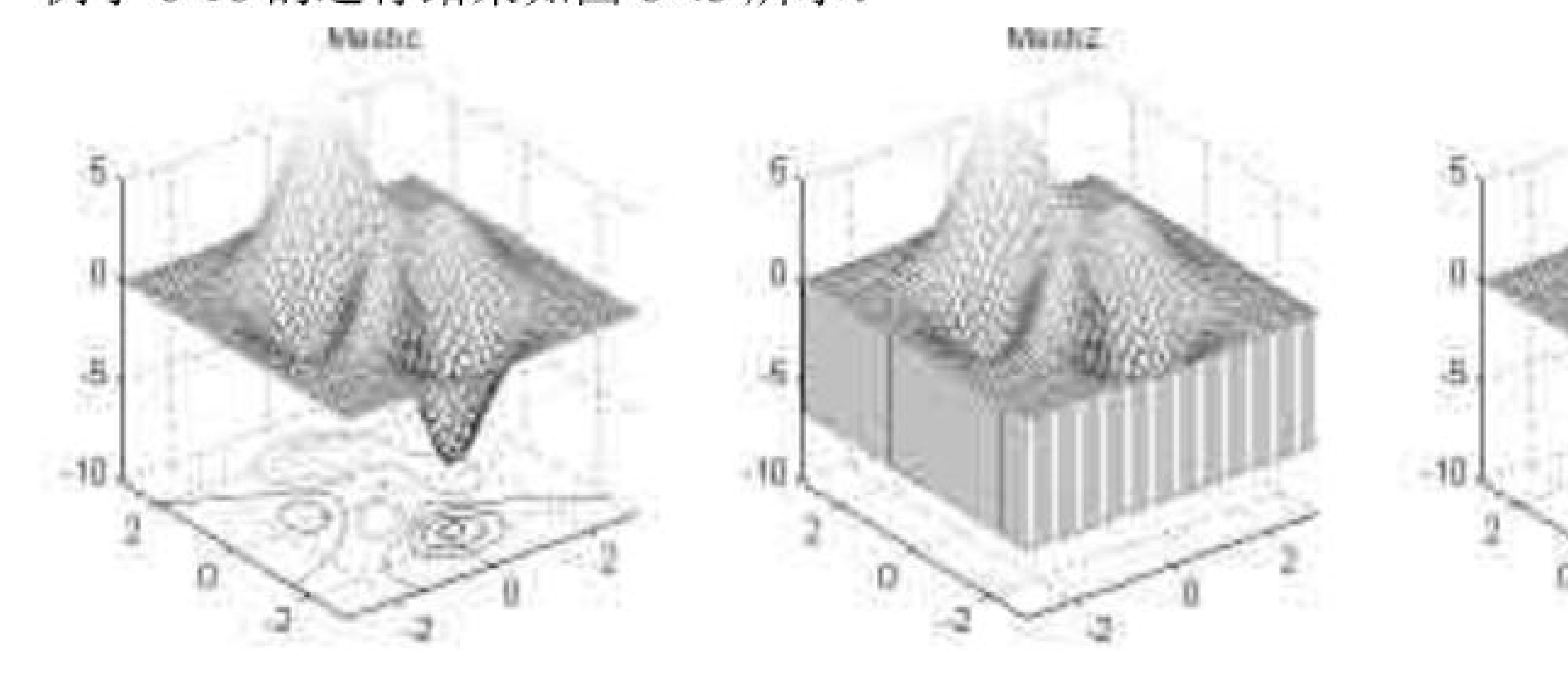


图 6-43　mesh 函数使用示例

在例子 6-16 中，使用了 meshgrid 函数创建二维的网格数据。meshgrid 函数可以用来创建三维曲线绘图的原始数据，它根据输入参数，创建等间距的网格数据。关于 meshgrid 函数的详细信息请参阅 MATLAB 的帮助文档或者函数的在线帮助。

通过例子 6-16 的运行结果能够明显地看出函数 mesh、mehsc、meshz 之间的区别。有关 mesh 函数的详细使用方法请参阅 MATLAB 的帮助文档。

surf 函数和 mesh 函数不同，surf 函数能够创建用色彩表示的曲面图，而不是线框图，而且该函数有一种变形，就是 surfc。所以，若将例子 6-16 的代码进行适当的修改，则得到的结果如图 6-44 所示。

surf 函数具体的使用请参阅 MATLAB 的帮助文档或者函数的在线帮助。其他的三维曲线、曲面绘制函数就不再赘述了。最后给出一个三维曲面的绘制示例，这里主要使用了 waterfall 函数、contour3 函数和前面介绍过的 plot3 函数。若对例子 6-17 使用的函数有所疑问，请参阅 MATLAB 的帮助文档。

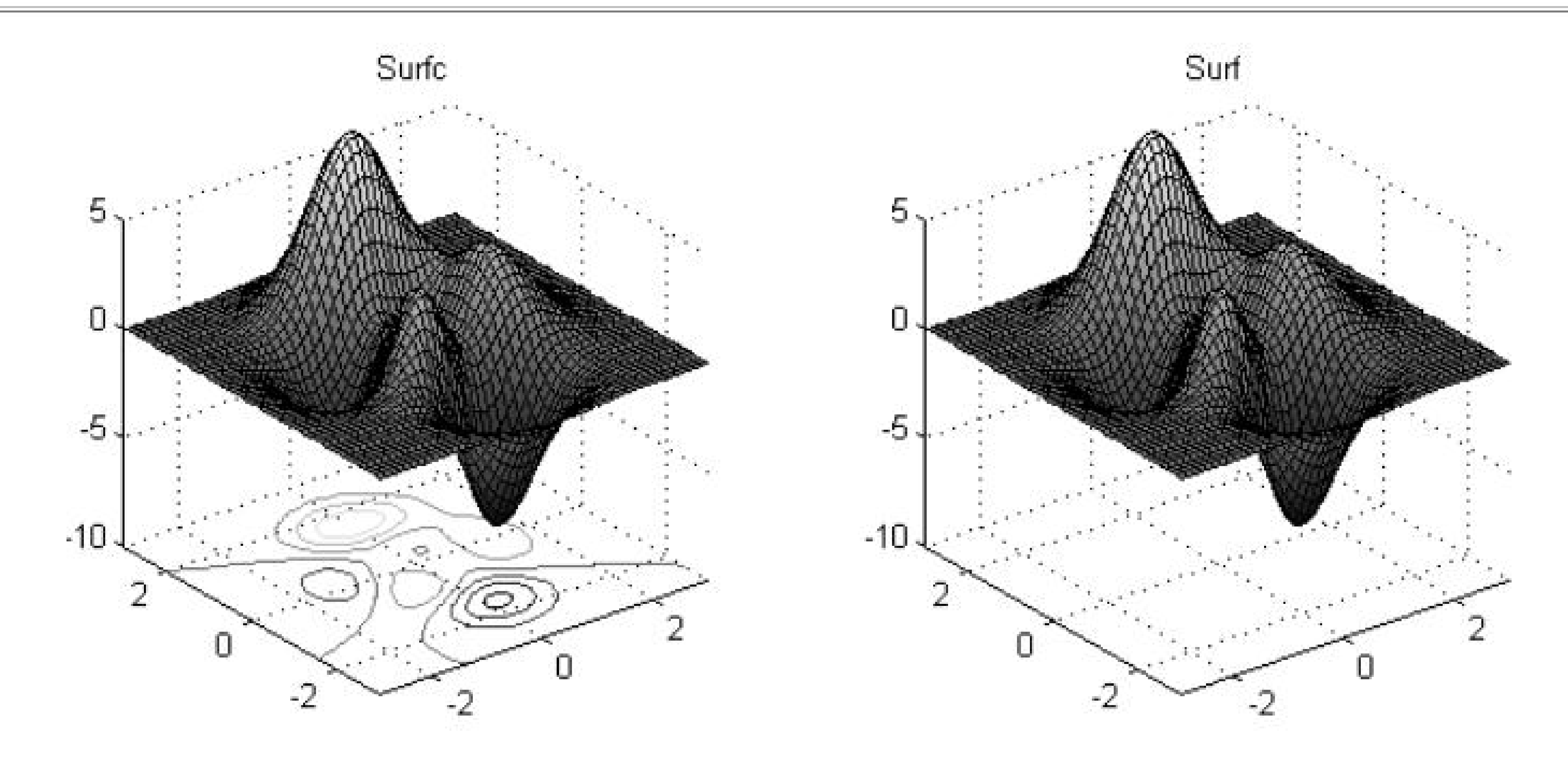


图 6-44　surf 函数使用示例

例子 6-17　其他三维绘图函数示例。

例子 6-17 的脚本文件代码如下：

```
%SURF_3D 三维绘图函数示例
% 准备数据
x = - 3:0.3:3; y = x;
[X,Y]=meshgrid(x,y);
[theat,R] = cart2pol(X,Y);
Z = sinc(R);
% 等高线
subplot(2,2,1)
contourf(peaks(30), 10)
colorbar
grid on
title('Peaks Function - (CONTOURF & COLORBAR)')
% plot3 函数绘制矩阵数据
subplot(2,2,2)
plot3(X,Y,Z)
grid on
axis([- 3 3 - 3 3 - 1 1])
title('Sinc Function - (PLOT3)')
% waterfall 函数，效果类似 surfz 函数
subplot(2,2,3)
waterfall(membrane(1));
title('L-shaped Membrane - (WATERFALL)')
%三维等高线
subplot(2,2,4)
```

```
contour3(peaks(30), 25);
title('Peaks Function - (CONTOUR3)')
colormap hsv
```

运行例子 6-17 的结果如图 6-45 所示。

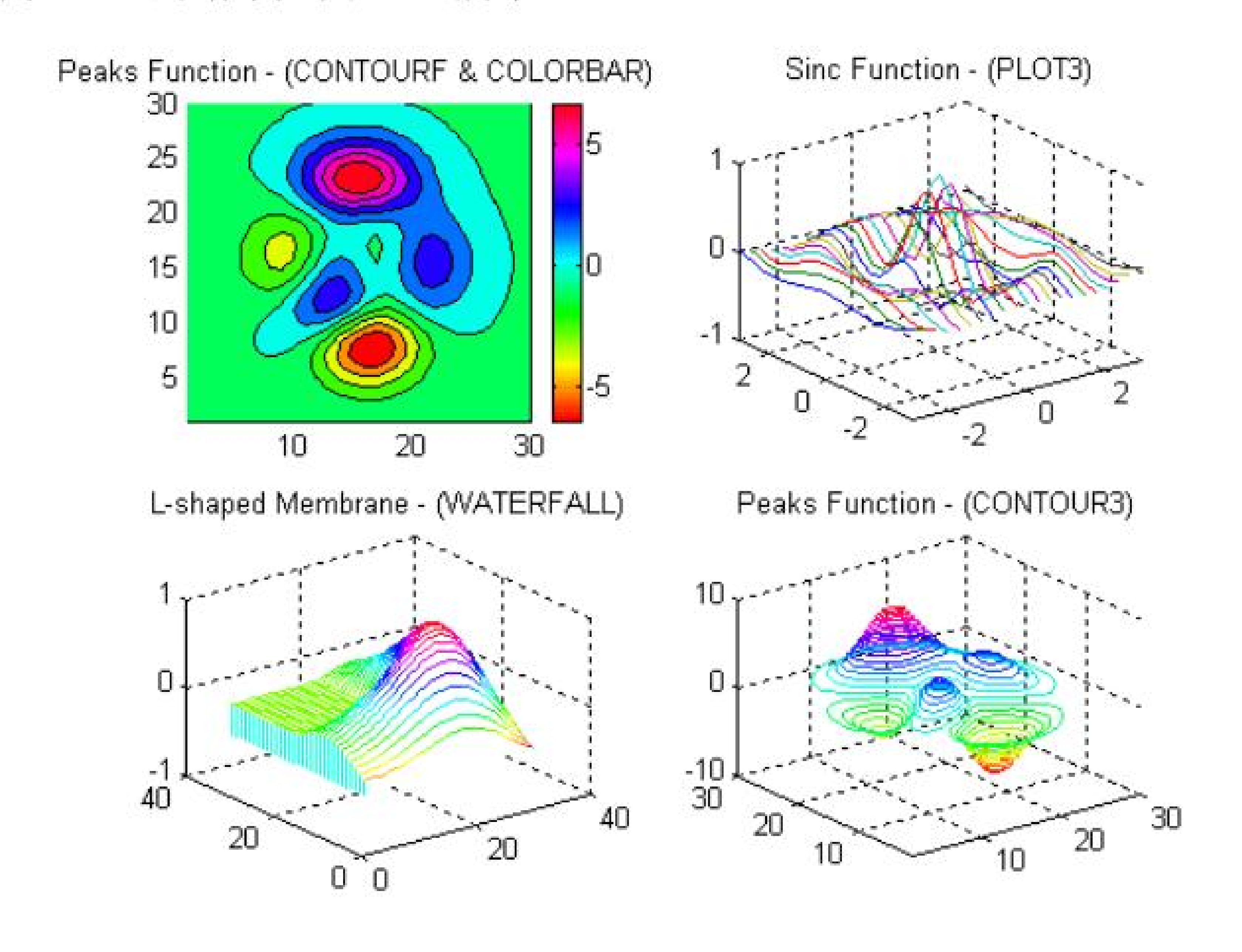


图 6-45　三维绘图函数的应用示例

6.6　保存和输出图形

为了便于使用绘制好的图形曲线，MATLAB 提供了将图形窗体中的内容输出到图形文件，或者将图形打印出来的功能。本小节将简要介绍一下将图形窗体中的内容导成图形文件和打印图形时需要使用的函数以及它们的注意事项。

6.6.1　保存和打开图形文件

MATLAB 支持将图形文件保存成为二进制格式的文件。为此，MATLAB 提供了一种类似于 MAT 格式的文件用来保存 MATLAB 的图形文件，这种文件的扩展名为*.fig。这种二进制的图形格式文件只能够在 MATLAB 中使用。

若需要将文件保存成为 fig 格式的图形文件，则在图形窗体中选择“File”菜单下的“Save”命令，或者直接单击工具栏上的保存按钮，在弹出的对话框中选择保存类型为 .fig，如图 6-46 所示。

在对话框中给定文件名称，然后单击“保存”按钮就可以保存文件了。

打开文件的过程和保存文件的过程类似，都可以通过菜单命令或者工具栏的按钮完成操作。

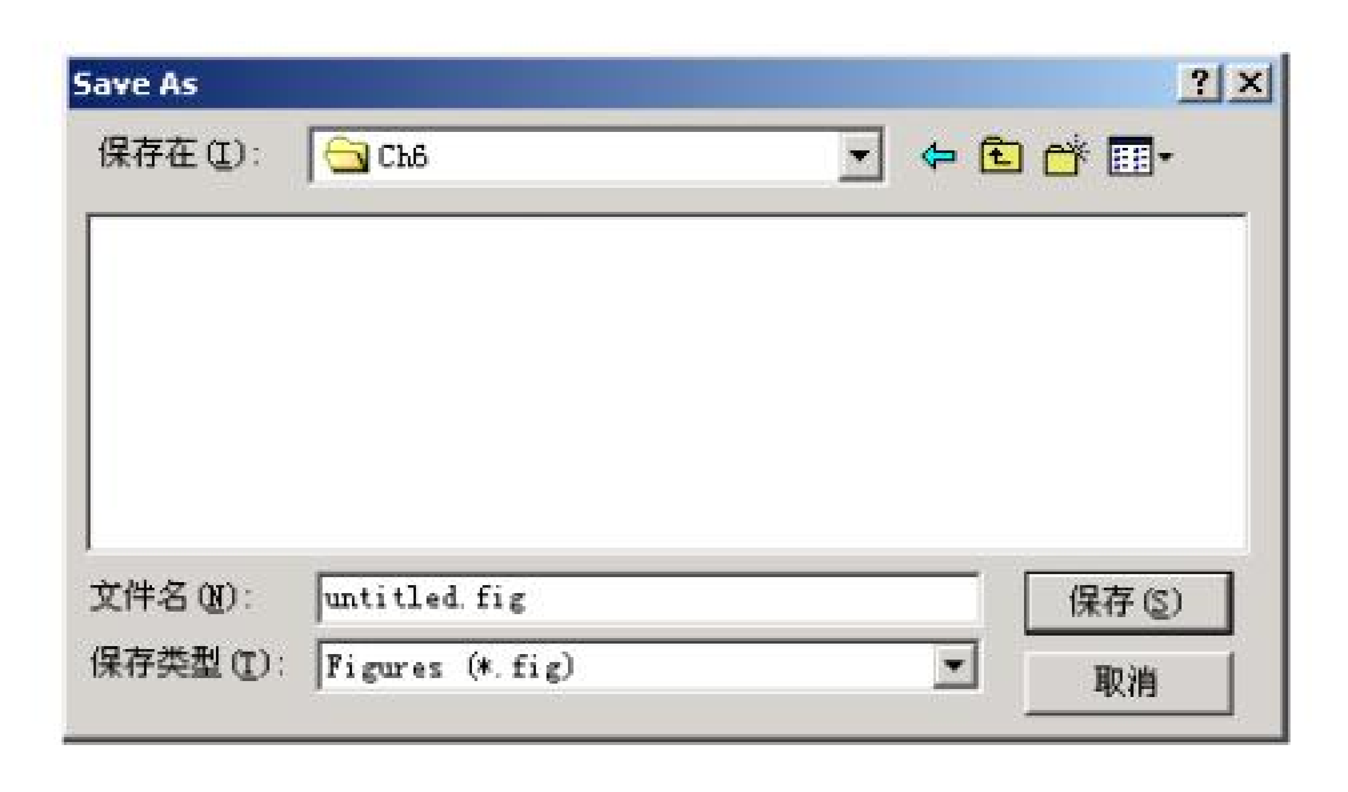


图 6-46　保存图形文件的对话框

不过，MATLAB 独具特色的就是为保存的图形文件提供了相应的命令，这个命令就是 saveas，该命令的一般语法结构如下：

```
saveas(h, 'filename.ext');
saveas(h, 'filneame','format');
```

其中，h 为图形的句柄，例如可以直接使用 gcf 指令获取当前的图形窗口的句柄；filename 为保存的文件名，而 saveas 命令根据 ext 的不同将文件存成为不同的格式。在第二种命令行格式中，format 直接说明文件的保存格式，它可以是图形文件的扩展名，或者是 m，或者 mfig，在取 m 或者 mfig 的时候，文件将被保存成为一个可调用的 M 文件和相应的图形数据文件。

打开图形文件就只要用 open 命令就可以了，open 函数会根据文件的扩展名不同而调用相应的辅助函数文件。例如在打开 .fig 图形文件时，调用 open 命令，具体过程请参阅例子 6-18。

例子 6-18　在命令行中保存打开图形文件。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> surf(peaks(30))
>> % 将图形文件保存为 M 文件和 fig 文件
>> saveas(gcf,'peakfile','M')
>> % 调用 M 文件重新显示窗体
>> peakfile
>> % 使用 open 指令打开文件
>> open('peakfile.fig')
```

在上面的短短几行代码中，代码将图形文件保存成为了一个 M 文件和一个 fig 文件，其中 M 文件的主要内容是：

```
function h = peakfile
[path, name] = fileparts(which(mfilename));
figname = fullfile(path, [name '.fig']);
if (exist(figname,'file')), open(figname), else open([name '.fig']), end
if nargout > 0, h = gcf; end
```

在文件中，不可缺少的就是大量的注释信息，而 M 文件的代码主要保证了可靠地打开保存的图形文件。最后的 open 命令能够完成同样的功能。

6.6.2　导出文件

尽管保存 .fig 文件非常方便，但是 .fig 文件却只能够在 MATLAB 中使用，所以 MATLAB 的图形窗口还可以将图形文件保存成其他的特殊图形格式文件。在表 6-3 中列举了能够直接在图形窗体中导出的图形文件类型。

表 **6-3**　**MATLAB** 支持的图形文件格式

文件类型	扩展名	文件类型	扩展名
增强型图元文件	emf	TIFF 图形文件	tif
位图	bmp	TIFF 格式非压缩文件	tif
EPS 文件	eps	便携式网络图像格式	png
EPS 色彩文件	eps	24 位位图文件	pcx
EPS 二级文件	eps	便携式位图	pbm
EPS 二级色彩文件	eps	便携式灰度图	pgm
Adobe Illustrator 文件	ai	便携式像素图	ppm
JPEG 图形文件	jpg		

若需要将图形文件保存成表 6-3 列举的各种类型文件，需要执行图形窗体“File”菜单下的“Export”命令，然后在对话框中选择需要导出的图形文件格式，最后给出相应的文件名，单击“保存”按钮后完成操作，对话框如图 6-47 所示。

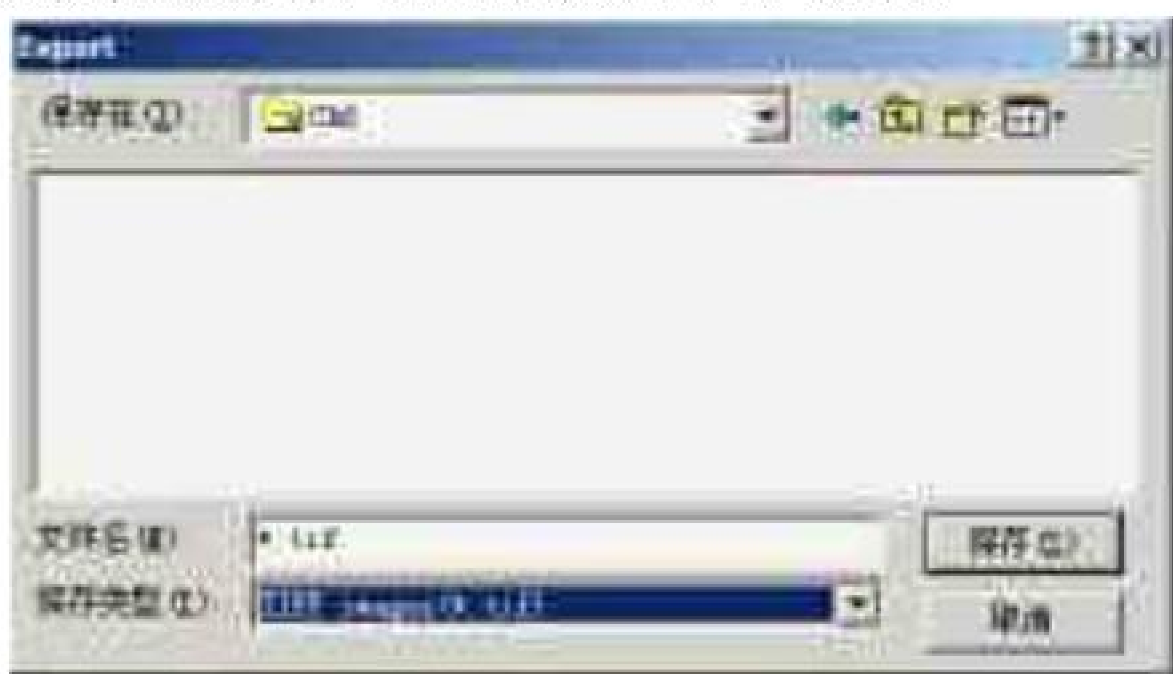


图 6-47　导出文件对话框

在前面小节介绍的 saveas 指令中，也可以使用这些扩展名来保存图形文件。例如将图形文件保存成为 tiff 格式的文件，命令行为

```
saveas(h,filename, 'tif');
```

MATLAB 提供了另外一个功能强大的命令来保存图形文件，这个命令就是 print 命令。

从直观上看，print 命令的作用是将图形文件通过打印机输出，它也支持将图形文件保存成其他格式的图形文件或者数据文件，这些图形文件不仅仅是那些在表 6-3 中列出的图形文件，而且还有更多的格式可以被支持，例如 PostScript 格式的文件等。

MATLAB 还提供了一个名为 printopt 的 M 文件，该文件主要可以由系统管理员编辑，以指明缺省的打印机类型和打印目标，当调用它时，返回缺省值的打印命令和设备选项，

例如在 Windows 系统下执行该命令：

```
>> [pcmd,dev] = printopt
pcmd =
COPY /B $filename$ $portname$
dev =
-dwin
```

print 命令的基本使用方法如下：

print -device -options filename

在命令行中，device 和 options 的选取有很多选项，而 device 可以是某种 MATLAB 支持的打印驱动，也可以是某种图形文件格式。由于受篇幅的限制这里就不一一解释说明了，请读者参阅 MATLAB 的帮助文档：doc print。

若将图形文件输出到打印机，则使用以下命令行：

```
>> print
```

这时 print 命令就可以直接将图形文件输出到当前系统默认的打印机。

将图形输出成为 PostScript 文件，可以指定相应的设备，例如：

```
>> print –dps filename
```

可以将图形文件保存为黑白的 PS 文件，而命令行：

```
>> print –dpsc filename
```

可以将图形文件保存为彩色的 PS 文件。

6.6.3　拷贝图形文件

在 Windows 操作系统中，可以将图形窗体中的内容拷贝到剪贴板上，然后可以将剪贴板上的内容拷贝到任何一个 Windows 应用程序中。拷贝图形时，不是简简单单地按下快捷键 Ctrl+C 就可以了，而是需要通过图形窗体“Edit”菜单下的“Copy Figure”命令来完成，如图 6-48 所示。

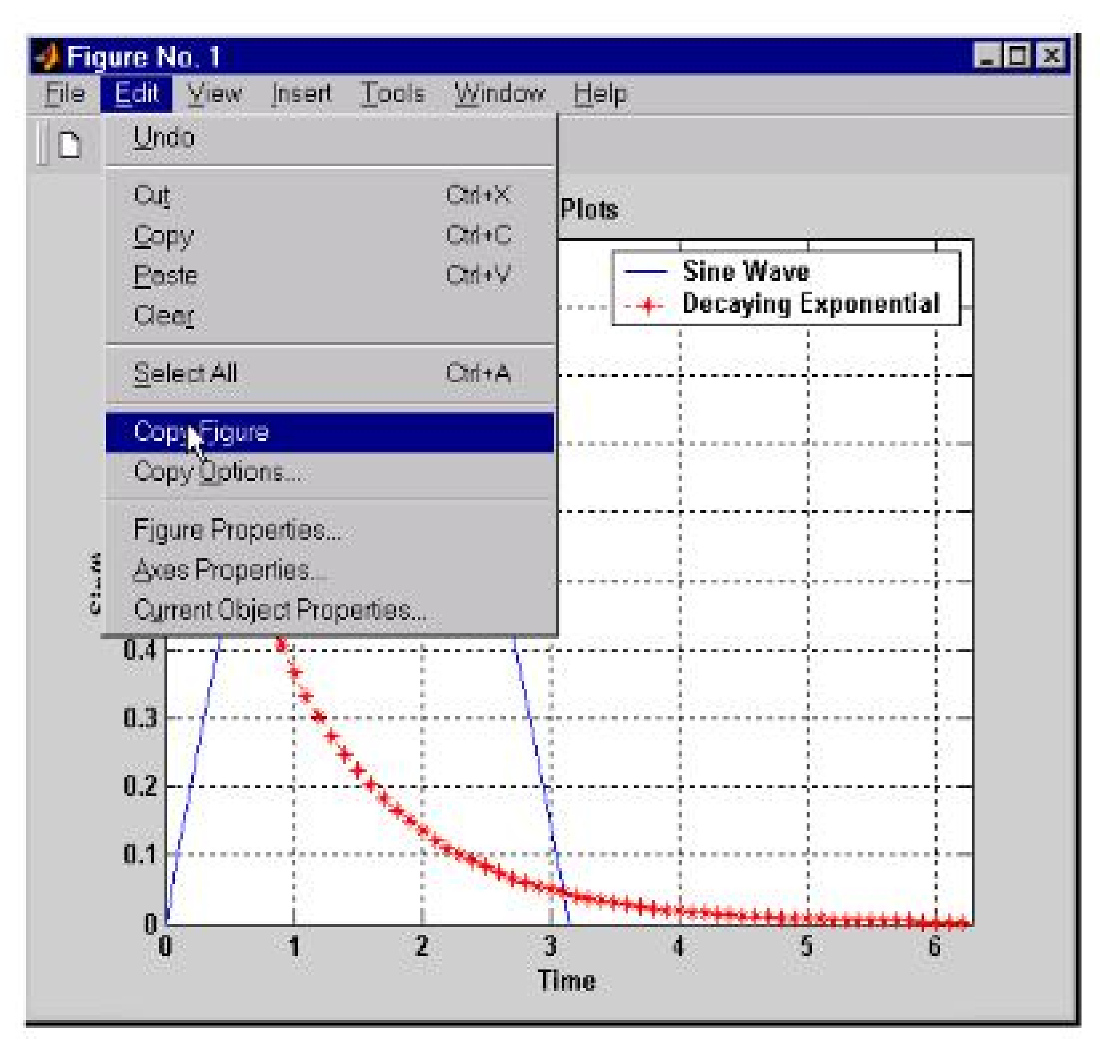


图 6-48　拷贝图形的菜单命令

这时拷贝下来的图形粘贴出来的效果类似前面章节图 6-24 的效果。可以通过选择“MATLAB”菜单下的“Reference”命令，在弹出的对话框中对控制拷贝图形效果的选项参数进行设置，如图 6-49 所示。

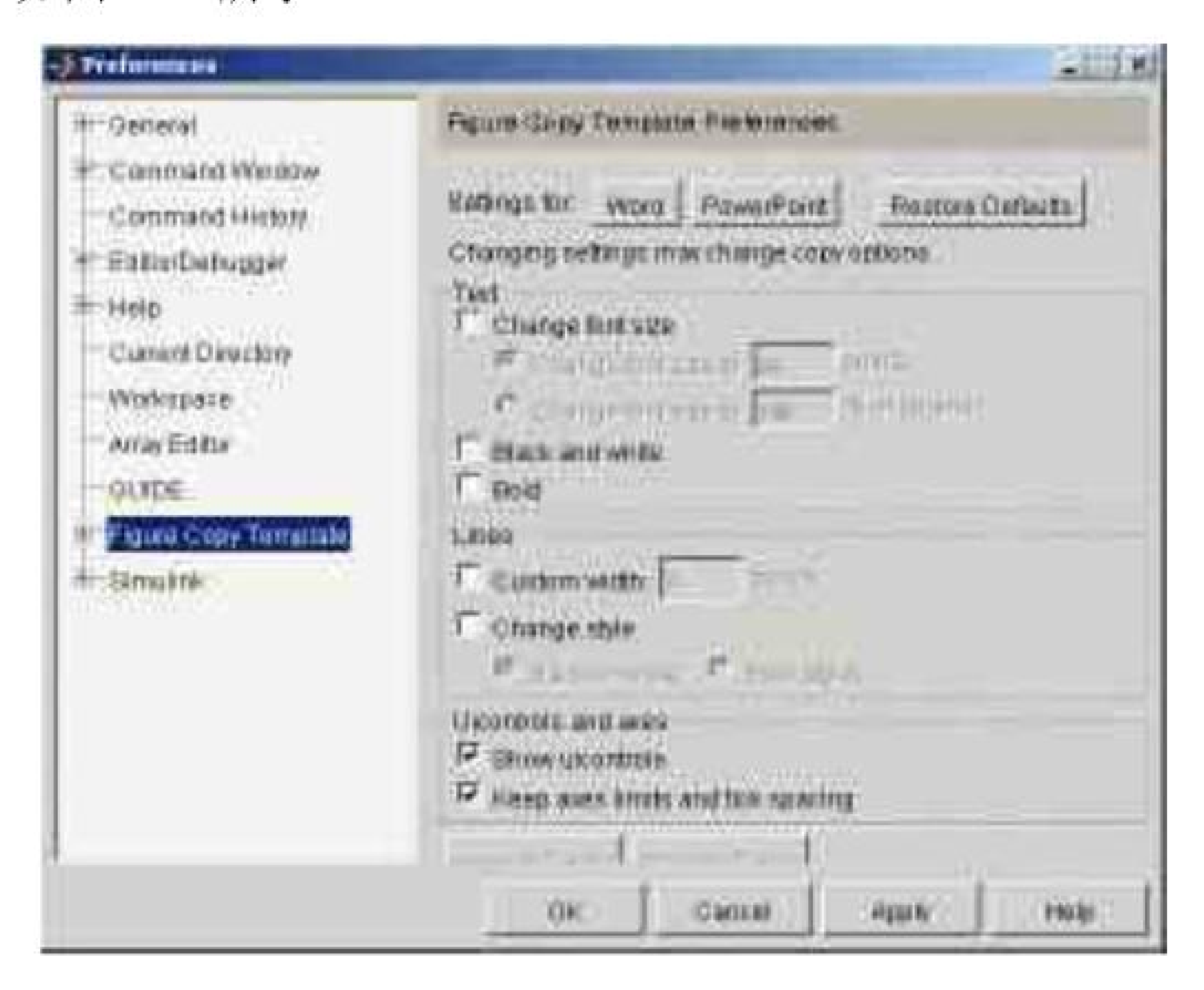


图 6-49　设置拷贝图形的模板

MATLAB 提供了两种 Windows 应用程序的拷贝模板——Word 和 PowerPoint，用户可以在如图 6-49 所示的对话框中设置相应的模板选项，而更加精细的设置需要在拷贝选项对话框中进行。

在如图 6-50 所示的对话框中可以设置精细的拷贝选项，比如剪贴板的选项以及背景图的设置等。当拷贝图形效果不甚理想的时候，可以通过修改这些设置来达到满意的效果。

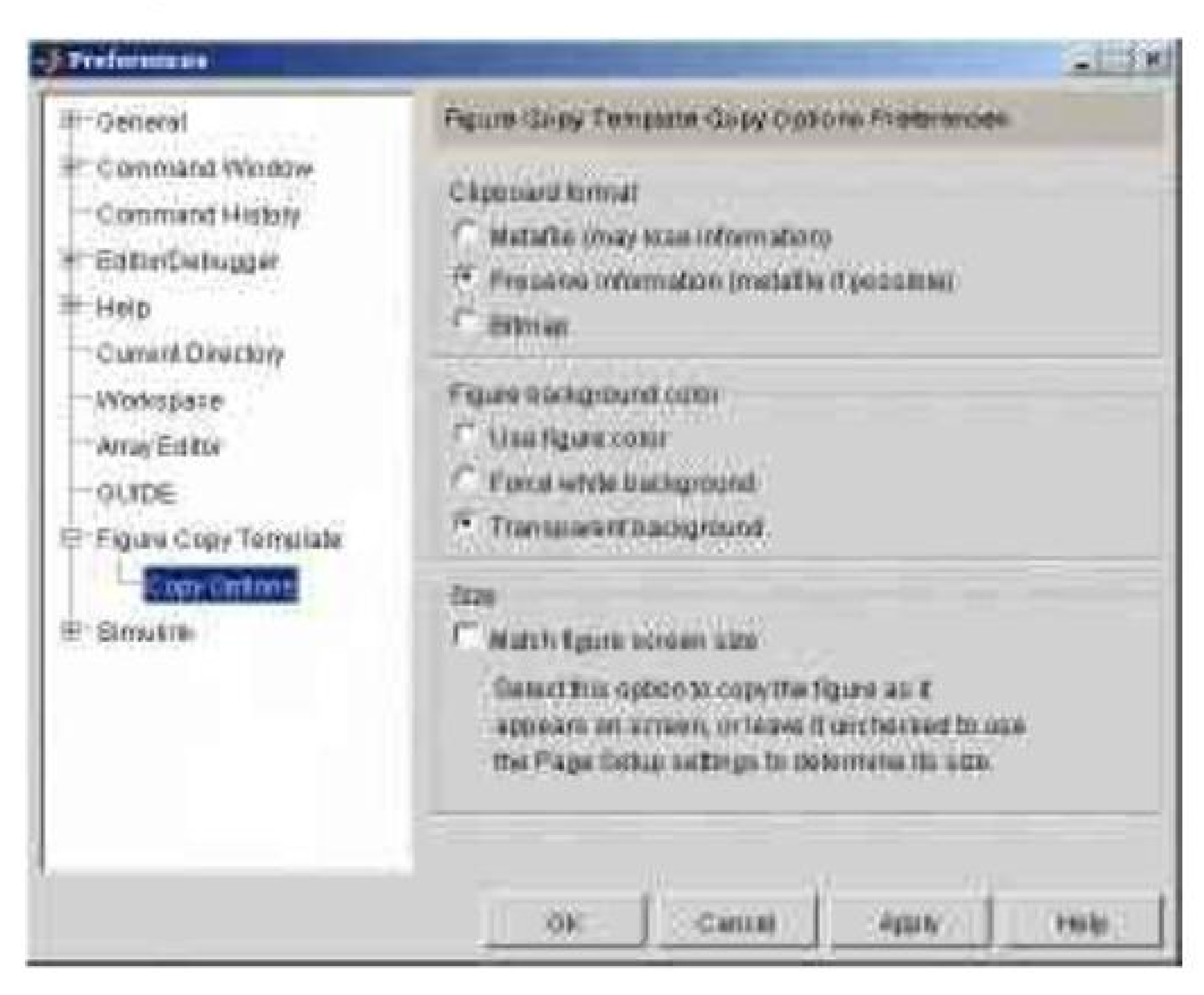


图 6-50　拷贝选项对话框

6.7　数据插值和曲线拟合

将数据绘制出来是手段，而处理数据是数据可视化的直接目的。在数据处理过程中，数据插值和曲线拟合是数据处理的基本手段。MATLAB 为数据插值和曲线拟合提供了最基本的方法，在 MATLAB R13 版本中，新增加了一个工具箱，名为 Curve Fitting Toolbox，这个工具箱提供了高级的曲线拟合方法。在本小节将简要介绍在 MATLAB 中进行数据插值和曲线拟合的方法。

6.7.1　插值运算

一般地，从各种试验得来的数据总是有一定的数量，而利用插值技术能够从有限的数据中获取系统整体的状态，因此，数据插值在各行各业，特别是信号处理领域内有着广泛的应用。MATLAB 软件作为数据处理的优秀软件之一，自然提供了常用的各种数据插值计算的函数。在本小节，主要介绍由 MATLAB 基本模块提供的插值计算函数。

MATLAB 进行插值计算的函数见表 6-4。

表 **6-4**　插值计算函数

函数	说明
interp1	一维插值(数值查表)
interp1q	一维快速插值(数值查表)
interp2	二维插值(数值查表)
interp3	三维插值(数值查表)
interpn	N 维插值(数值查表)
interpft	使用 FFT 算法的一维插值
griddata	二维数据网格的表面数据插值
griddata3	三维数据网格的超表面(hypersurface)数据插值
griddatan	多维数据网格的超表面数据插值
mkpp	产生分段多项式
pchip	分段的厄密多项式
ppval	计算分段多项式的数值
spline	三次杨条插值
unmkpp	分段多项式的细节

在表 6-4 的函数中较常用的就是进行基本插值的 interp 系列函数，其中 interp1 和 interp1q 函数比较起来，后者在处理 X 为单调递增的向量，Y 为列向量或者行数同 X 向量长度的矩阵进行插值计算的时候，速度较快，而且仅能进行线性插值。

interp1 函数一般的用法为

yi = interp1(x, y, xi, method)

其中：

- x 和 y 为原始数据。
- xi 为需要计算的插值点。
- method 可以为插值计算指定相应的算法，为字符串类型，其取值可以为 nearest、linear、spline、cubic、pchip、v5cubic。

在 interp 系列函数中 method 参数使用的几种不同的取值分别对应了不同的插值计算方法，例如 linear 为线性插值算法，它也是系统默认的插值算法，而 spline 为三次杨条插值算法。有关插值算法请参阅相应的数值分析教科书，对于更高次的插值算法函数，请参阅 MATLAB 的帮助文档。

若进行插值运算时，xi 的取值超过了 x 的范围，则需要进行外插运算，这个时候需要在使用函数的时候指定参数 extrap，即函数的使用方法为

```
yi = interp1(x, y, xi, method, 'extrap');
```

例子 6-19　一维插值函数示例。

例子 6-19 使用的脚本文件如下：

```
%INTERP_EX1 一维插值计算示例
% 准备数据
x = 0:10;
y = cos(x);
% 插值点
xi = 0:0.2:10;
% 进行插值运算
yin = interp1(x,y,xi,'nearest');
yic = interp1(x,y,xi,'cubic');
% 绘制结果
plot(x,y,'o',xi,yin,'*',xi,yic)
legend('origin','nearest','cubic')
title('一维插值计算示例')
```

运行脚本文件得到的结果如图 6-51 所示。在例子 6-19 的代码中，使用了两种一维插值算法，分别为 nearest 和 cubic，由于 nearest 算法仅计算插值点左近的数值，所以从结果上看插值得到的结果很不理想，而三次插值计算得到的结果相对要理想得多。

不过从计算时间和内存等系统资源的损耗上考虑，nearest 是最快的而又最节省资源的一种算法，而三次插值或者样条插值运算则需要消耗较多的系统资源，所以请用户根据需要选择合适的插值算法。

二维插值运算函数的使用方法类似一维插值运算函数的使用方法，同样也可以在使用函数的同时指定相应的算法：nearest、linear、cubic 和 spline 等。与一维插值不同的是，这里使用的插值算法都需要进行双次运算，所以也可以在算法选择时使用 bilinear 或者 bicubic 关键字指定相应的算法。在例子 6-20 中，对这几种插值算法进行了比较。

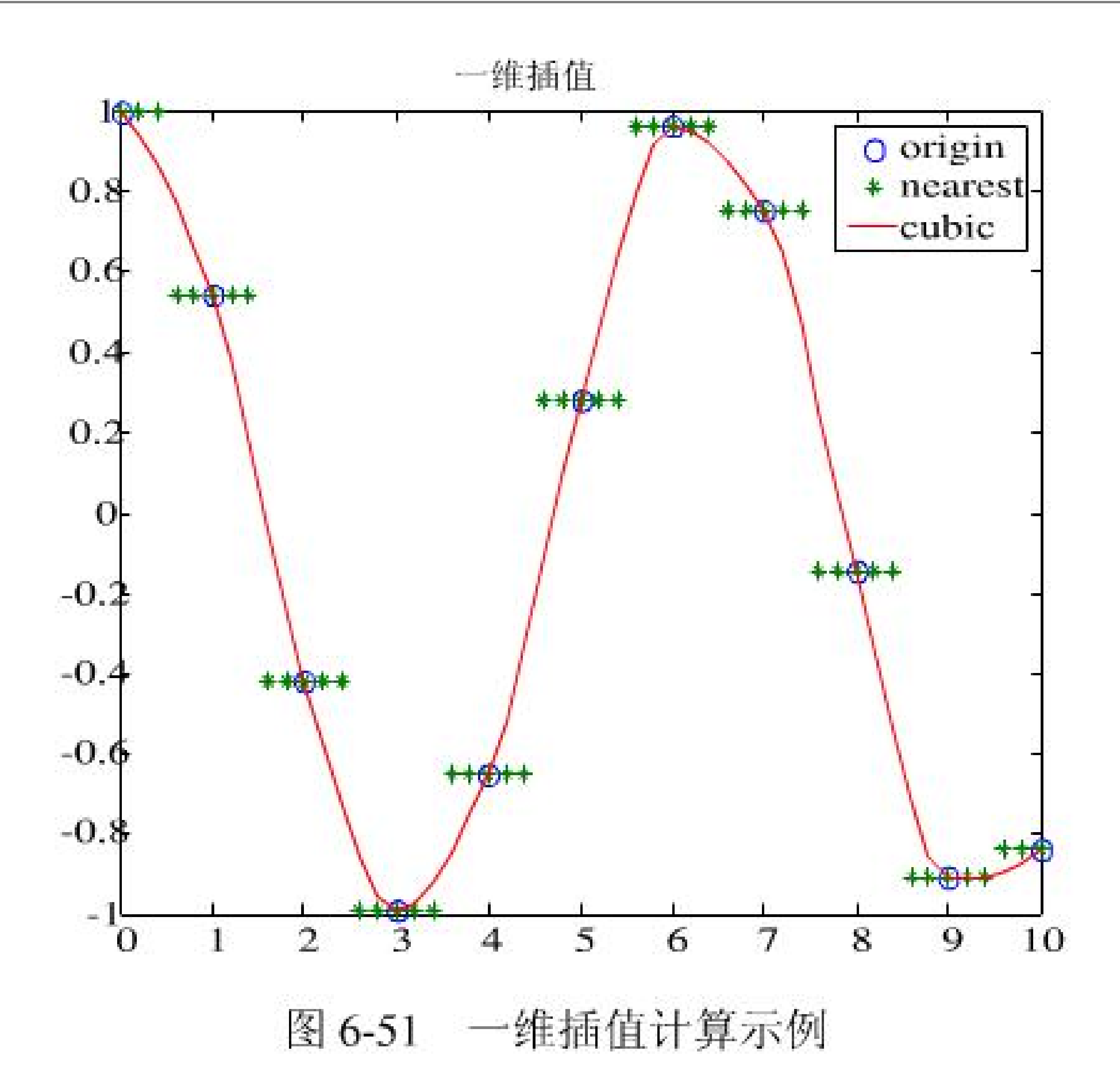


图 6-51　一维插值计算示例

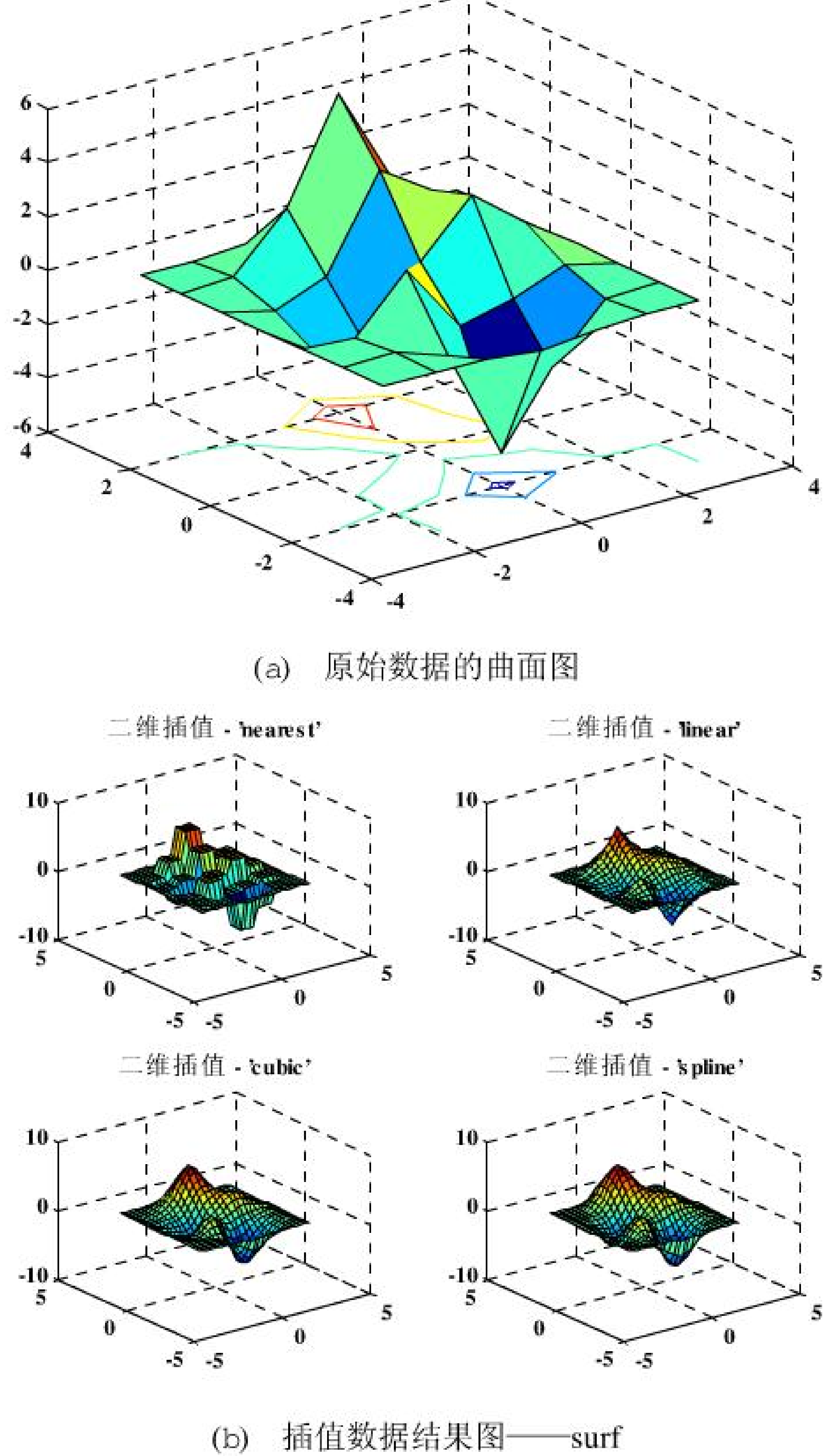
例子 6-20　二维插值运算算法比较。

```
function compare_interp( )
%COMPARE_INTERP 不同插值运算的比较

% 原始数据
[x,y] = meshgrid(- 3:1:3);
z = peaks(x,y);
figure(1); clf
surfc(x,y,z);
title('原始数据')
% 进行插值运算
[xi, yi] = meshgrid(- 3:0.25:3);
zi1 = interp2(x,y,z,xi,yi,'nearest');
zi2 = interp2(x,y,z,xi,yi,'linear');
zi3 = interp2(x,y,z,xi,yi,'cubic');
zi4 = interp2(x,y,z,xi,yi,'spline');
% 通过可视化结果比较
figure(2)
subplot(2,2,1);surf(xi,yi,zi1);
title('二维插值 - "nearest"')
subplot(2,2,2);surf(xi,yi,zi2);
title('二维插值 - "linear"')
subplot(2,2,3);surf(xi,yi,zi3)
title('二维插值 - "cubic"')
subplot(2,2,4);surf(xi,yi,zi4)
title('二维插值 - "spline"')
% 可视化结果
```

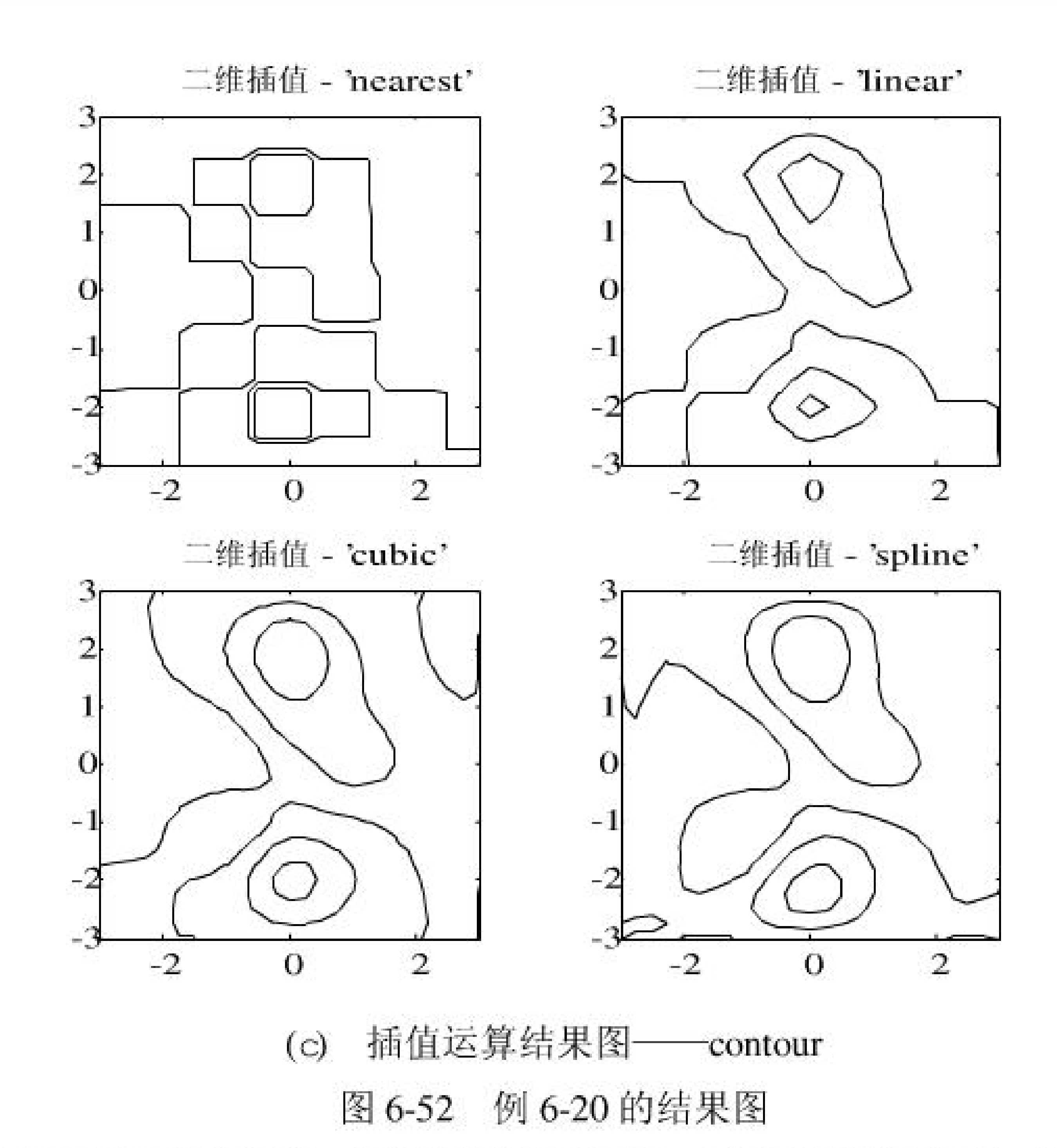
```
figure(3)
subplot(2,2,1);contour(xi,yi,zi1)
title('二维插值 - "nearest"')
subplot(2,2,2);contour(xi,yi,zi2)
title('二维插值 - "linear"')
subplot(2,2,3);contour(xi,yi,zi3)
title('二维插值 - "cubic"')
subplot(2,2,4);contour(xi,yi,zi4)
title('二维插值 - "spline"')
```

在运行例子 6-20 的代码时，可以打开 MATLAB 的性能分析器(profiler)，这样便于比较不同算法消耗的系统时间。例子 6-20 的结果如图 6-52(a)~(c)所示。



(a)　原始数据的曲面图

(b)　插值数据结果图——surf



(c)　插值运算结果图——contour

图 6-52　例 6-20 的结果图

若在执行例子 6-20 的代码时，打开性能分析器，则可以看到 012～014 行代码占用的计算时间，下面为分析文件的片段：

```
0 060          2      nt r 2(        n  r  t)
0 0 0              2  nt r 2(         n  r)
0 05                  nt r 2(        cu  c)
0  00          5      nt r 2(            n  )
```

通过例子 6-20 的运算结果可以看出，不同的插值算法在运算结果和运算消耗时间上的差别。用户同样需要根据自己的需要选择不同的算法。

有关数据插值的应用还有很多内容，受篇幅的限制这里就不再一一详细介绍。有兴趣的读者可以通过学习有关数值分析的知识，同时了解和掌握不同的 MATLAB 数据插值函数。

6.7.2　曲线拟合

曲线拟合的任务和数据插值的任务不同，曲线拟合需要从一些离散的数据中推导出两者之间的数学解析关系，而数据插值是通过原始数据计算一些新的离散数据点。曲线拟合的结果一般为一个或者多个数学解析关系，利用这些解析关系能够对数据进行一定的推断，准确的曲线拟合结果可以用来进一步评估、验证实测的数据。在本小节将介绍利用 MATLAB 的基本模块进行曲线拟合的函数和方法，而 MATLAB 高级的曲线拟合应用请参阅 MATLAB 和 Curve Fitting 工具箱的帮助文档。

利用 MATLAB 进行曲线拟合主要有两种方法：回归法拟合和多项式拟合，本小节通过一些具体的示例来说明这两种不同的曲线拟合方法。

例子 6-21　回归法曲线拟合。

回归法主要使用 MATLAB 的左除运算来寻找曲线拟合解析函数的系数。例如有这样一组数据：

```
>> t = [0 .3 .8 1.1 1.6 2.3]';
>> y = [0.5 0.82 1.14 1.25 1.35 1.40]';
```

将这组数据绘制出来：

```
>> plot(t,y,'r*')
>> grid on
```

得到如图 6-53 所示的图形结果。

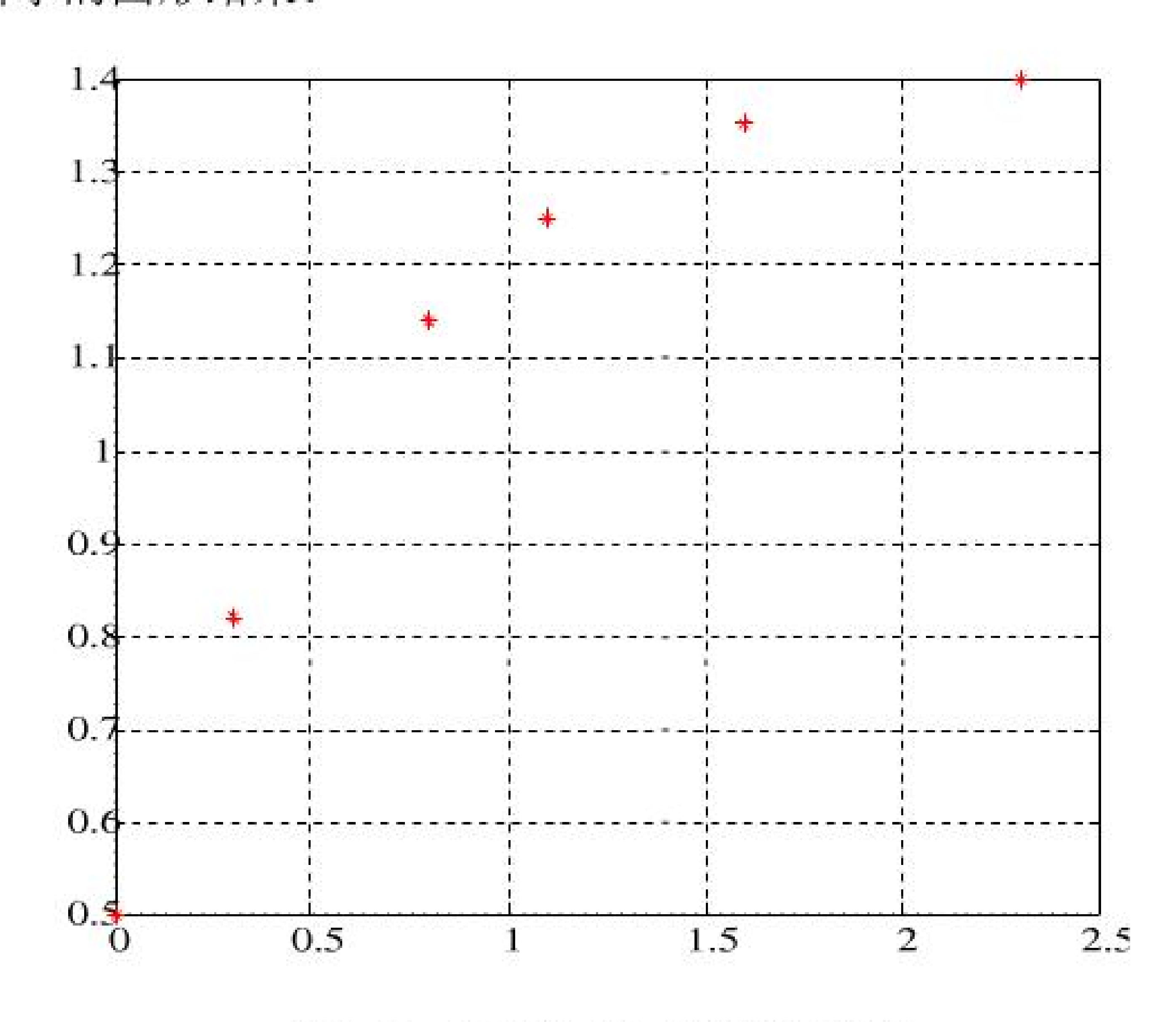


图 6-53　进行拟合计算的原始数据

通过图 6-53 的分布情况，可以猜测该数据由如下的表达式得出：

$$y = a_0 + a_1 t + a_2 t^2$$

进行曲线拟合的任务就是分别求得表达式中的 a_0、a_1、a_2 三个系数，这样就可以得到相应的等式关系：

$$Y=AT$$

若需要求得 A 则只要计算除法：

$$A=Y/T$$

于是，在 MATLAB 中键入下面的指令，就可以得到计算的结果：

```
>> X = [ones(size(t))   t   t.^2]
X =
      1.0000         0         0
      1.0000    0.3000    0.0900
      1.0000    0.8000    0.6400
      1.0000    1.1000    1.2100
```

```
        1.0000      1.6000      2.5600
        1.0000      2.3000      5.2900
>> A = X\y
A =
        0.5318
        0.9191
       - 0.2387
```

这样得到的多项式应该为

$$y=0.5318+0.9191t-0.2387t^2$$

为了验证结果，可以进一步进行计算：

```
>> T = (0:0.1:2.5)';
>> Y = [ones(size(T))    T    T.^2]*A;
>> plot(T,Y,'-',t,y,'o'), grid on
>> legend('Fitting','Origin')
```

这时得到的图形结果如图 6-54 所示。

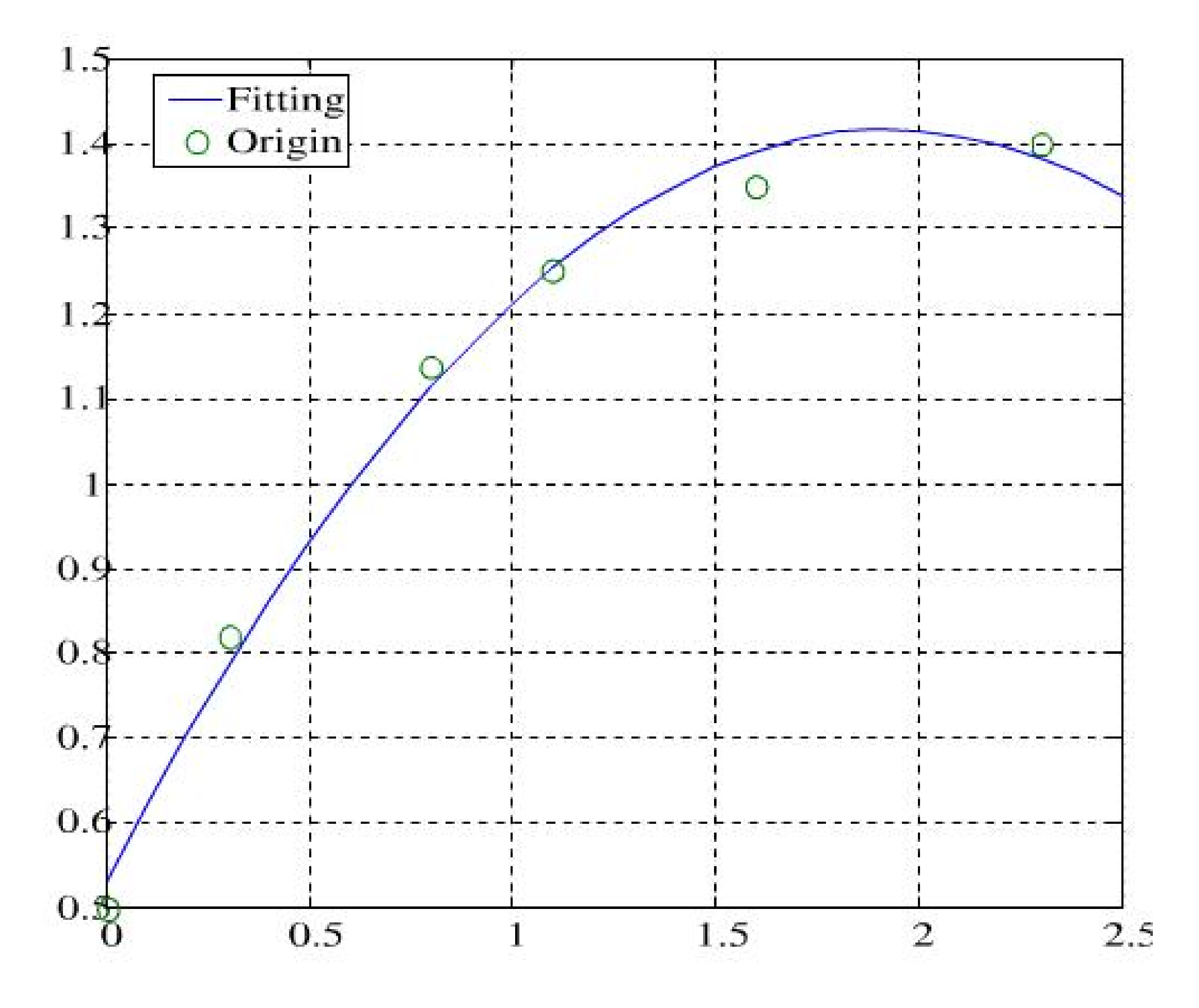


图 6-54　拟合结果与原始数据的比较

除了多项式以外，还可以猜测原始数据的多项式是由指数函数组成的：

$$y=a_0+a_1e^{-t}+a_2te^{-t}$$

于是，为了求得系数，可以在 MATLAB 的命令行中键入下面的指令：

```
>> X = [ones(size(t))    exp(- t)    t.*exp(- t)]
X =
        1.0000      1.0000           0
        1.0000      0.7408      0.2222
```

```
        1.0000    0.4493    0.3595
        1.0000    0.3329    0.3662
        1.0000    0.2019    0.3230
        1.0000    0.1003    0.2306
```

左除：

```
>> A = X\y
A =
     1.3974
    - 0.8988
     0.4097
```

这样得到的多项式应该为

$$y=1.3974-0.8988e^{-t}+0.4097te^{-t}$$

为了验证这次得到的结果，可以进一步计算：

```
>> T = (0:0.1:2.5)';
>> Y = [ones(size(T))    exp(-T)    T.*exp(-T)]*A;
>> plot(T,Y,'-',t,y,'o'), grid on
>> legend('Fitting', 'Origin')
```

这时得到的曲线图形如图 6-55 所示。

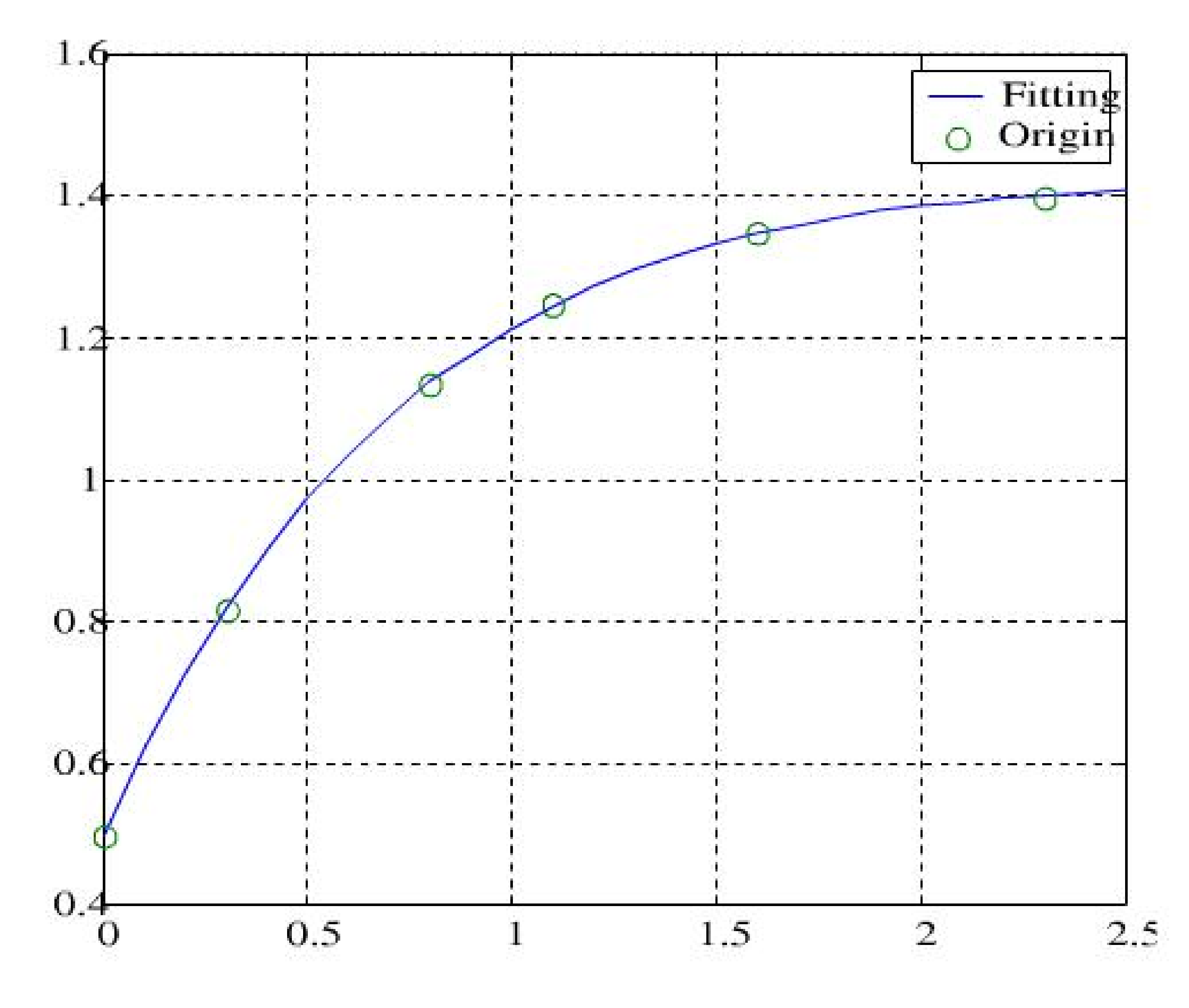


图 6-55　曲线拟合结果和原始数据比较

可以看出，在两次的曲线拟合结果中，利用指数函数拟合的结果比较好。

另外一种曲线拟合的方法是多项式拟合。在 MATLAB 中，进行多项式拟合主要使用两个函数：polyfit 和 polyval。

polyfit 函数主要用来进行拟合计算，它的基本语法为

```
p = polyfit(x,y,n)
```

其中，x 和 y 为参与曲线拟合计算的原始数据，n 为进行拟合计算的多项式次数，函数的返回值是多项式的系数，也就是说，函数的运算结果为多项式的系数向量。

$$y = p_n x^n + p_{n-1} x^{n-1} + \ldots + p_1 x + p_0$$

polyval 函数主要用来计算多项式的数值，它的基本语法为

```
y = polyval(p,x)
```

其中，p 为多项式的系数，而 x 是变量的数值，得到的结果就是函数的数值向量。

例子 6-22　polyfit 函数和 polyval 函数的应用示例。

下面是例子 6-22 的脚本文件：

```
001    %CURVE_FIT 多项式拟合计算示例
002    % 准备原始数据
003    x = 0:.1:10;
004    y = sin(x)+cos(2*x);
005    % 5 次多项式拟合
006    k5 = polyfit(x,y,5);
007    y5 = polyval(k5, x);
008    % 11 次多项式拟合
009    k11 = polyfit(x,y,11);
010    y11 = polyval(k11, x);
011    % 绘制数据曲线
012    plot(x, y, 'g', x, y5, 'r', x, y11, 'b')
013    % 标注
014    title('Curve Fitting', 'Fontsize',14)
015    legend('Original Curve', '5th order', '11th order',4 )
016    set(findobj('Type', 'line'), 'LineWidth', 2)
```

运行例子 6-22 的代码：

```
>> curve_fit
Warning: Polynomial is badly conditioned. Remove repeated data points
         or try centering and scaling as described in HELP POLYFIT.
(Type "warning off MATLAB:polyfit:RepeatedPointsOrRescale" to suppress this warning.)
> In E:\MATLAB6p5\toolbox\matlab\polyfun\polyfit.m at line 75
  In D:\Temp\Ch6\curve_fit.m at line 9
```

出现上述警告信息的原因是使用了过高的数据拟合阶数。由于 polyfit 函数是通过左除算法获取多项式系数的，如果此时矩阵接近奇异(不可逆)，但又需要对该矩阵进行求逆计算时，则会出现上述的警告信息。

例子 6-22 运行的图形结果如图 6-56 所示。

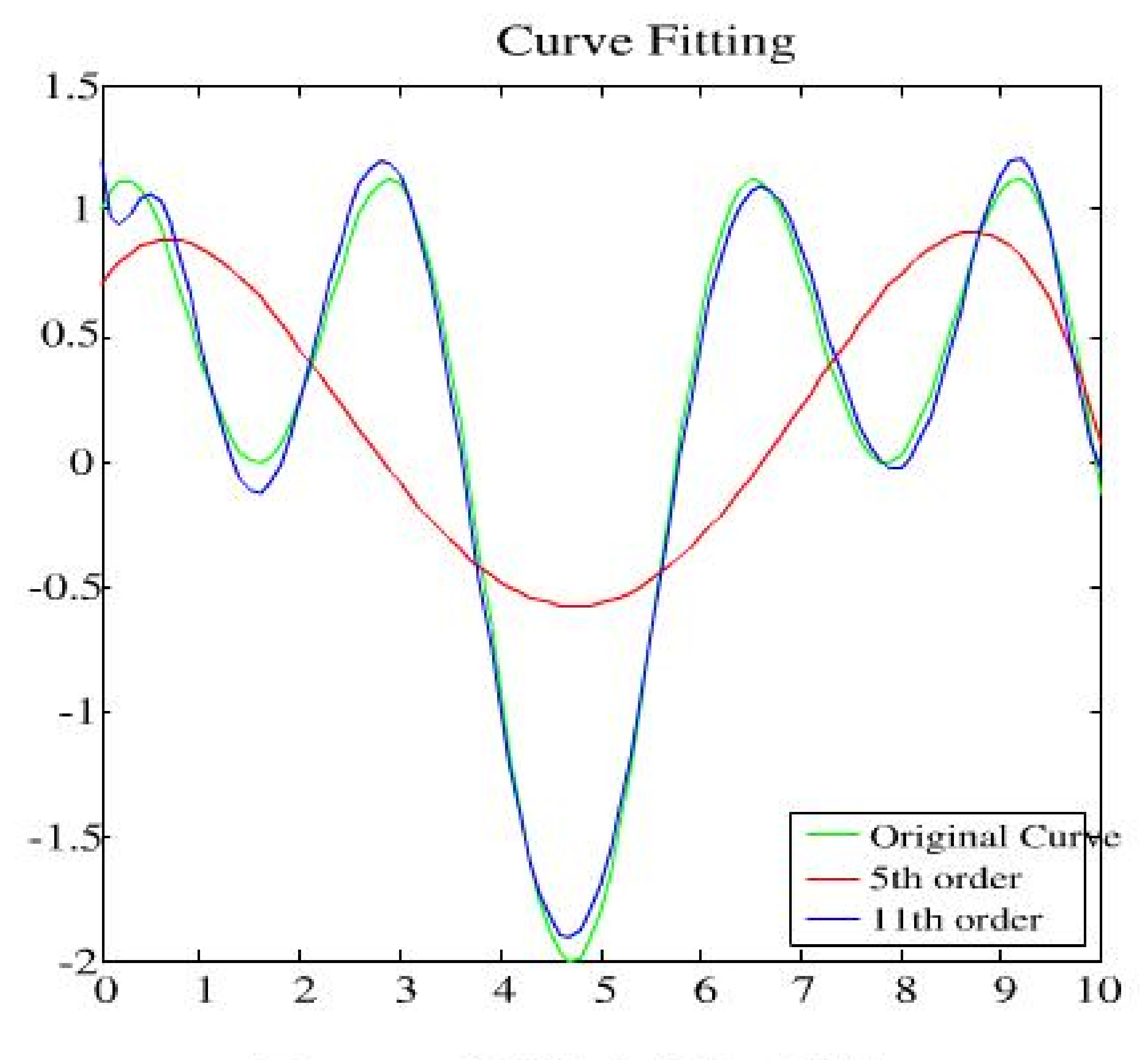


图 6-56　曲线拟合的图形结果

6.7.3　基本拟合工具

为了便于进行曲线拟合计算，MATLAB 提供了基本的曲线拟合工具界面，利用这个图形界面，可以方便地实现常用的曲线拟合工作，利用该界面可以完成下列工作：

- 使用三次样条曲线(cubic spline)或最高 10 阶的多项式拟合数据。
- 对一组给定的数据同时绘制多条拟合曲线。
- 绘制拟合残差曲线。
- 查看拟合的数值结果。
- 对拟合曲线求值(内插或外推)。
- 用拟合数值结果和残差的范数标注图形。
- 把拟合的结果保存到 MATLAB 工作区。

可以通过命令行函数和基本曲线拟合界面双管齐下地进行曲线拟合计算，不过，基本曲线拟合界面仅能针对二维数据进行拟合计算。

本小节通过一个具体的示例来说明基本曲线拟合工具的使用方法和步骤。

例子 6-23　使用基本曲线拟合工具。

进行曲线拟合工作的第一步是加载原始数据，在本例子中，使用的数据来自于 MATLAB 自带的 Demo，关于该 MATLAB 自带例子的信息请参阅在线帮助：help census。

在 MATLAB 命令行窗口中键入下面的指令：

```
>> load census
>> plot(cdate,pop,'ro');
```

绘制出原始数据后，执行图形窗体“Tools”菜单下的“Basic Fitting”命令，这时将弹出基本曲线拟合工具的图形用户界面，如图 6-57 所示。

在弹出的对话框中，首先选择需要进行曲线拟合的数据，在“Select data”的下拉列表框中，选择“data1”，即当前进行曲线拟合运算的数据。若坐标轴上有多条数据时，则可以选择不同的曲线拟合原始数据。




图 6-57　基本曲线拟合工具界面

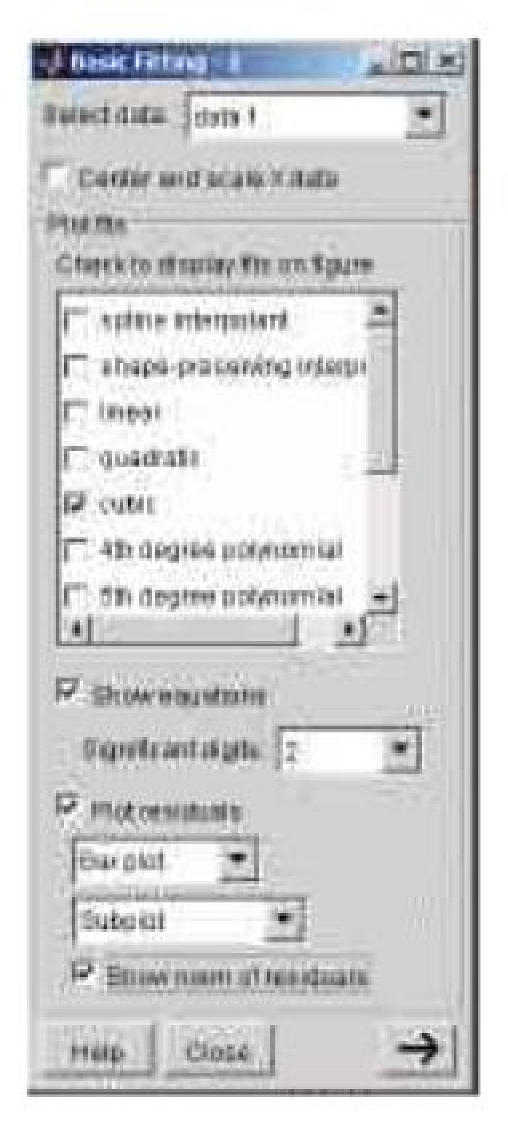


图 6-58　设置曲线拟合的属性

然后选择一个合适的拟合算法。在基本曲线拟合算法列表框中，选择“cubic”复选框，这时将使用三次多项式进行曲线拟合计算。选择“Show equations”复选框，则系统将曲线拟合的多项式绘制在图形中。另外，还可以通过选择不同的复选框，在图形窗体中输出曲线拟合的残差，以及残差的范数等，这时的曲线拟合工具界面如图 6-58 所示。

在选择曲线拟合算法的时候，系统有时候会给出如图 6-59 所示的对话框提示。



图 6-59　基本拟合计算的提示信息

该信息在前面利用函数进行多项式计算的时候已经有所提示，为了解决该问题，需要选择“Center and Scale X data”复选框。在选择不同的曲线拟合属性的同时，系统将不断地将曲线拟合结果输出在图形窗体中，在完成了上述操作步骤之后，曲线图形窗体的结果如图 6-60 所示。

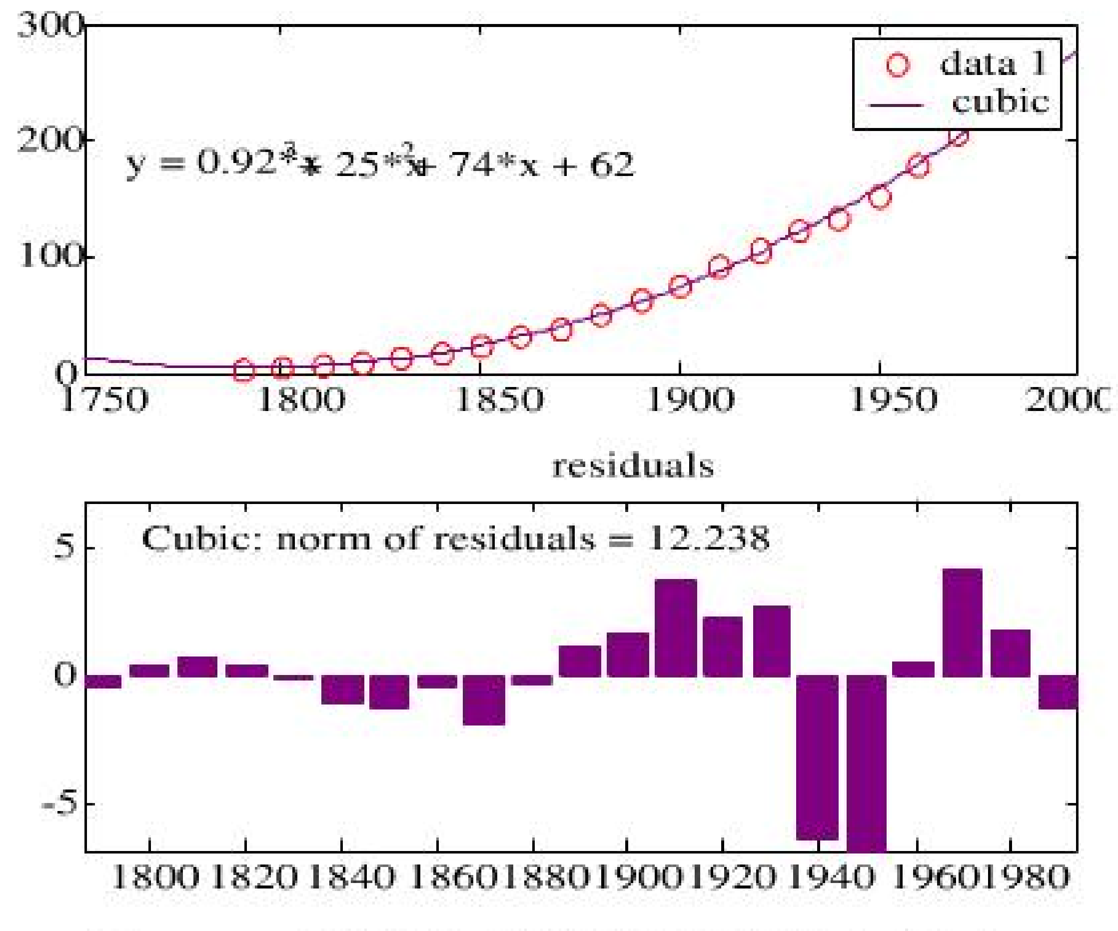


图 6-60　曲线拟合的结果绘制在图形窗体中

此时图形窗体中的注释是由系统自动添加的，可以使用图形编辑器修改其中的内容。

单击基本曲线拟合工具的➔按钮，可以得到基本曲线拟合工具的扩展，在这部分扩展的工具界面中，可以查看曲线拟合计算的结果，包括多项式的系数和范数等，如图 6-61 所示。

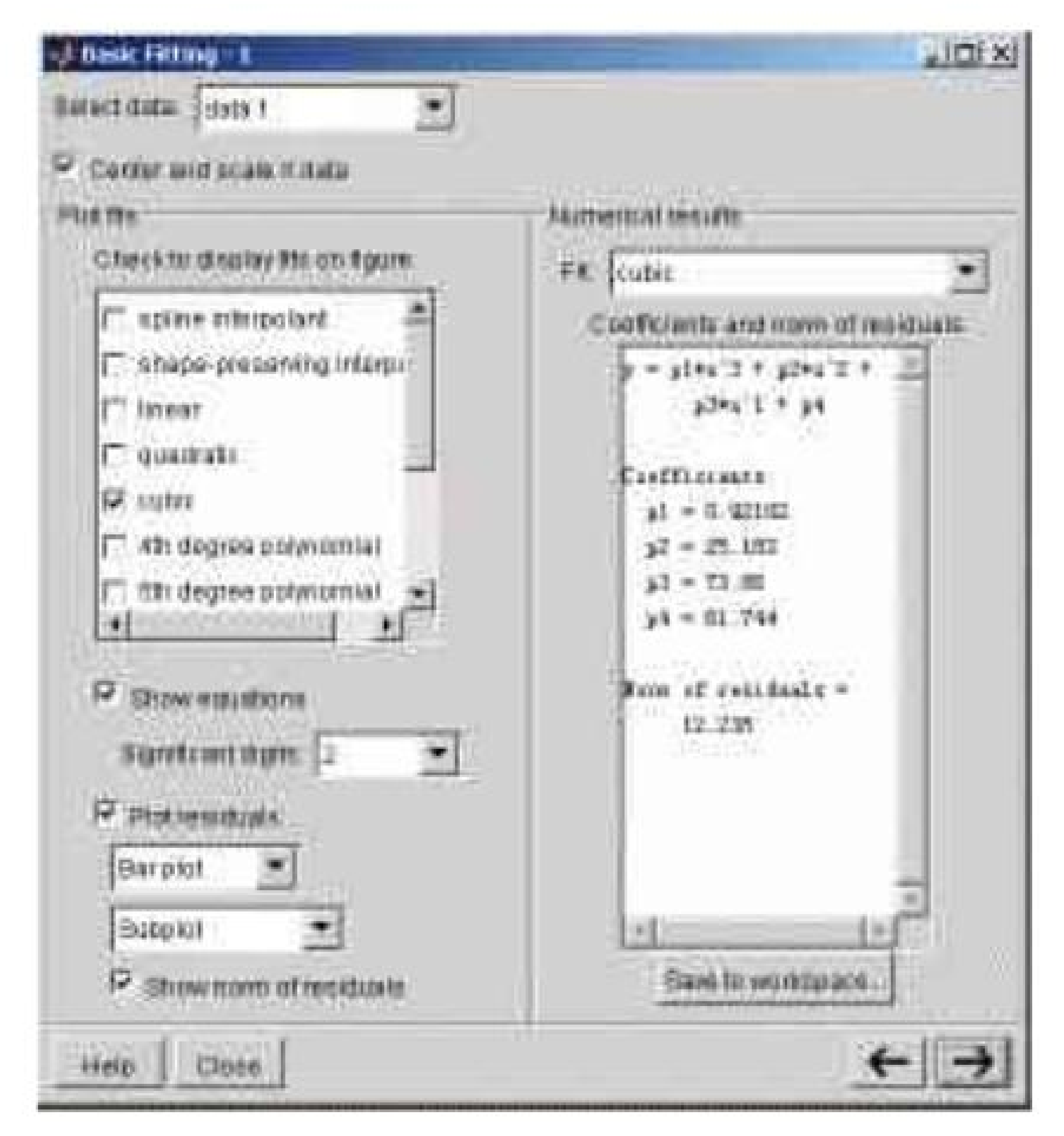


图 6-61　基本曲线拟合工具的界面扩展

在扩展界面中，单击“Save to workspace”按钮可以将曲线拟合计算得到的结果保存到工作空间中，如图 6-62 所示。

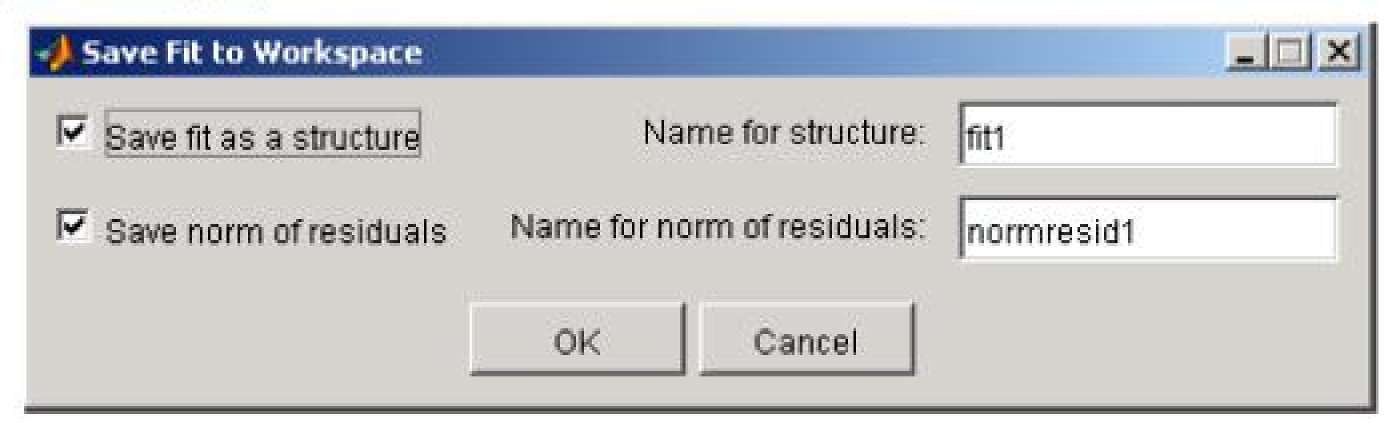


图 6-62　将曲线拟合的结果保存到工作空间

这时保存的结果为一个结构，按下“OK”按钮之后在 MATLAB 命令行窗口中输入下列指令进行查看：

```
Basic Fitting GUI created variables in the current workspace.
>> whos
  Name            Size                 Bytes  Class
  cdate           21x1                   168  double array
  fit1            1x1                    318  struct array
  normresid1      1x1                      8  double array
  pop             21x1                   168  double array
Grand total is 68 elements using 662 bytes
```

```
>> fit1
fit1 =
        type: 'polynomial degree 3'
       coeff: [0.9210 25.1834 73.8598 61.7444]
```

在工作空间中的变量包含了曲线拟合的多项式系数结果。利用该结果可以通过 polyval 函数进一步计算多项式的数值。

若再次单击基本曲线拟合界面上的→按钮，则能够得到再次扩展的工具界面，在该界面中可以计算一下当前拟合多项式的数值，如图 6-63 所示。

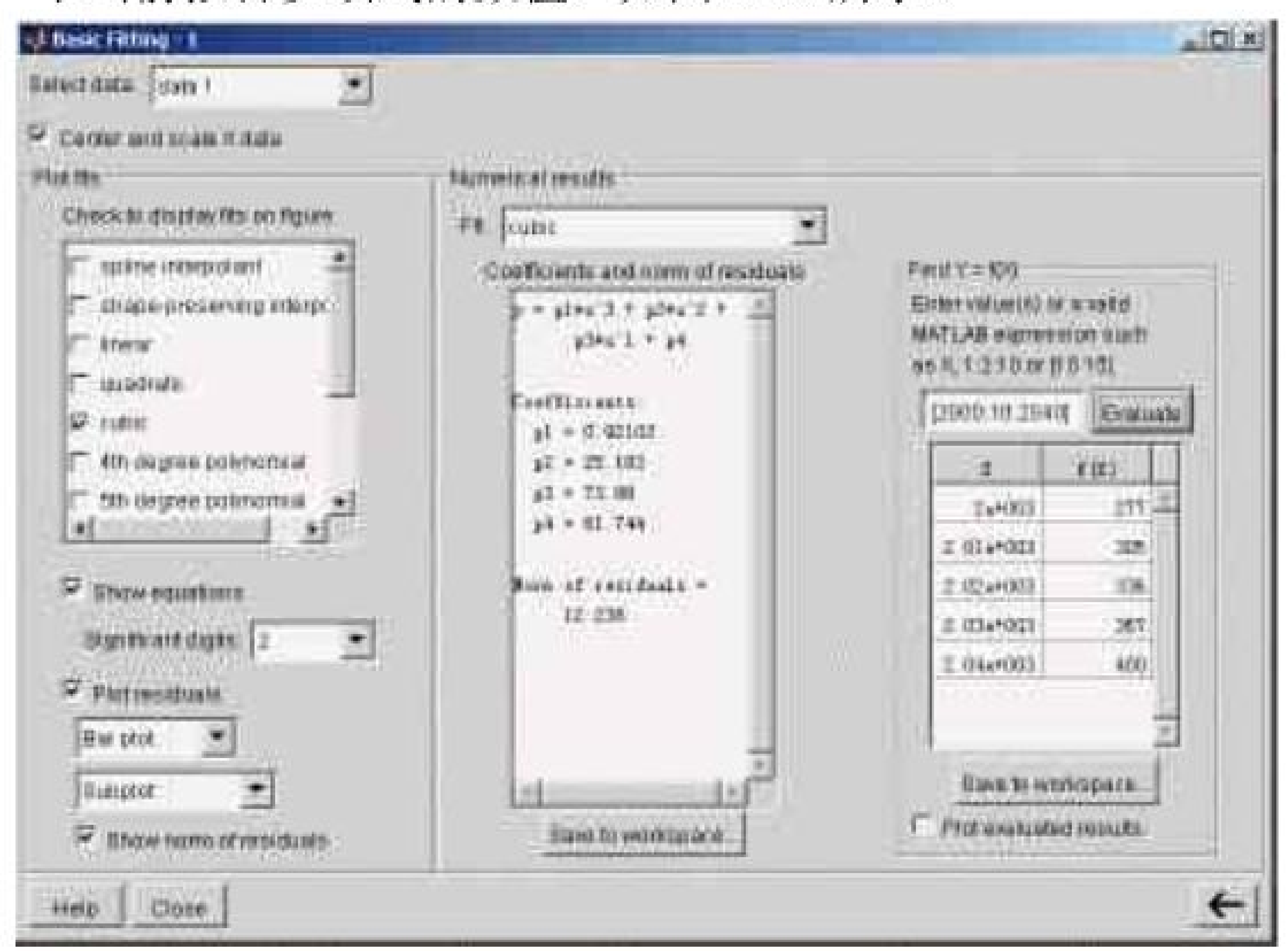


图 6-63　进一步扩展基本曲线拟合界面

在本例子中，进一步计算了 2000 年至 2040 年之间的人口数量的变化，间隔为 10 年。通过选择“Plot evaluated results”复选框，可以将这些拟合计算的结果也绘制在图形界面中，如图 6-64 所示。

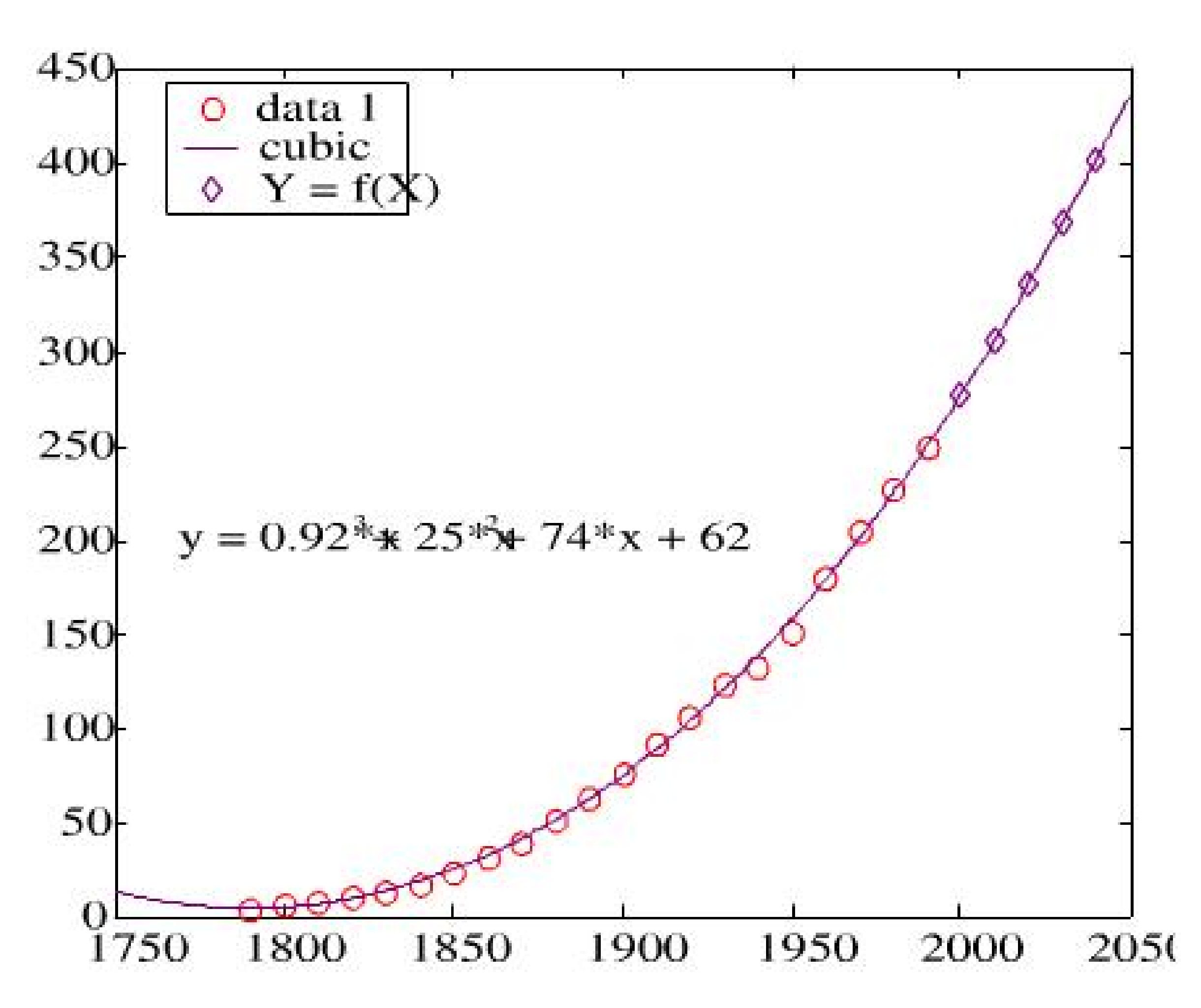


图 6-64　曲线拟合运算的结果

如果选择“Save to workspace”按钮，还可以将结果保存在工作空间中。

通过例子 6-23 的学习，读者应该已经基本掌握了基本曲线拟合工具的使用方法。更高级的曲线拟合工作可以使用 MATLAB 的曲线拟合工具箱来完成。不过有关曲线拟合工具箱方面的知识已经超出了本书的讨论范围，有兴趣的读者请参阅 MATLAB 的帮助文档。

6.8 本 章 小 节

在本章详细介绍了 MATLAB 的基本图形和数据可视化的能力。MATLAB 的图形和数据可视化是其强大功能的一个重要表现，这其中包括了灵活的二维和三维数据可视化、各种图形标注和特殊图形的绘制能力，另外 MATLAB 还提供了丰富的导入、导出数据图形的能力。MATLAB 不仅能够将图形绘制出来，而且还具有完成一定的数据分析的能力，在本章介绍了数据插值和曲线拟合的方法。通过本章的学习，读者应该能够掌握基本的数据可视化和绘图函数的使用，同时能够完成简要的数据统计分析的工作。除了利用 MATLAB 本身的数据分析功能以外，还可以利用 Curve Fitting 工具箱完成复杂的曲线拟合工作，使用 Statistics 工具箱进行高级数据统计工作。数据可视化和分析是科学计算、信号处理、控制系统应用等领域的重要内容，掌握本章的内容可以为这些领域内的高级应用打下坚实的基础。

本章介绍的函数和各种操作相对比较复杂，希望读者能够在自己日常工作中多使用这些功能，掌握其中的精髓。

第七章 GUIDE 入门

前面一章重点介绍了 MATLAB 强大而又灵活的数据图形可视化的功能。而 MATLAB 图形功能强大之处还体现在它可以创建图形用户界面。创建图形用户界面的方法主要有两种：图形句柄和 GUIDE。由于图形句柄体系较为复杂，而且编程的工作量比较大，所以在本章就不着重介绍图形句柄，而是将精力集中在介绍GUIDE的使用和图形用户界面编程上。有关图形用户界面和图形句柄高级应用方面的知识，有兴趣的读者可以参阅 MATLAB 的帮助文档。

本章讲述的主要内容如下：

- 图形句柄入门；
- □□□□□的使用；
- 图形用户界面实例。

提示：

GUIDE 为 Graphical User Interfaces Development Environment 五个单词的缩写。

7.1 概 述

MATLAB 作为一种科学计算软件，其基本的功能需要通过 M 语言编程来实现。那么通过图形用户界面的形式来发布应用程序的好处就是可以允许程序的使用者不具备很深厚的 MATLAB 知识或者数学知识，只要用户熟悉了解计算机的基本操作就可以完成相应的计算。MATLAB 的图形用户界面同流行的操作系统——Windows、Unix 或者 Linux 的图形界面类似，它使用这些平台上的统一外观作为自己的外观样式，它的图形用户界面应用程序可以做到一处编写到处运行，只要相应的平台上具有 MATLAB 即可。

在 MATLAB 中创建图形用户界面的方法有两种——图形句柄和 GUIDE，这两种实现的方法都需要使用 M 语言编程，但是技术的侧重点不同。其实 GUIDE 创建图形用户界面的基础也是图形句柄对象，只不过是具有很好的封装，使用起来简便，而且还能够做到可视化的开发，对于一般的用户使用 GUIDE 创建图形用户界面应用程序已经足够了。MATLAB 提供了基本的用户界面元素，包括菜单、快捷菜单、按钮、复选框、单选框、文本编辑框、静态文本、下拉列表框、列表框等。需要注意的是，MATLAB 的图形用户界面程序大多数是对话框应用程序，利用 MATLAB 编写文档视图应用程序相对来说比较困难。

使用 GUIDE 和图形句柄创建的图形用户界面的主要区别在于，利用图形句柄创建的图形界面应用程序只有一个文件——M 文件，而利用 GUIDE 创建的图形用户界面应用程序一般由两个文件组成，一个是应用程序文件——M 文件，另一个是外观文件——fig 文件。

MATLAB 图形用户界面的例子非常多，不仅在 MATLAB 的 Demos 中有很多用户界面的例子，如图 7-1 所示，在 MATLAB 的工具箱中也有很多是利用 GUIDE 编写的小工具，不过早期的 MATLAB 工具多数都使用图形句柄进行开发。若 MATLAB 的图形用户界面功能不能够满足用户的需要，用户还可以利用 Java 语言的工程来扩充界面功能，不过使用 Java 语言来扩充 MATLAB 的功能属于 MATLAB 外部接口编程的内容，有兴趣的读者可以参阅 MATLAB 的帮助文档或者《MATLAB 外部接口编程》一书。

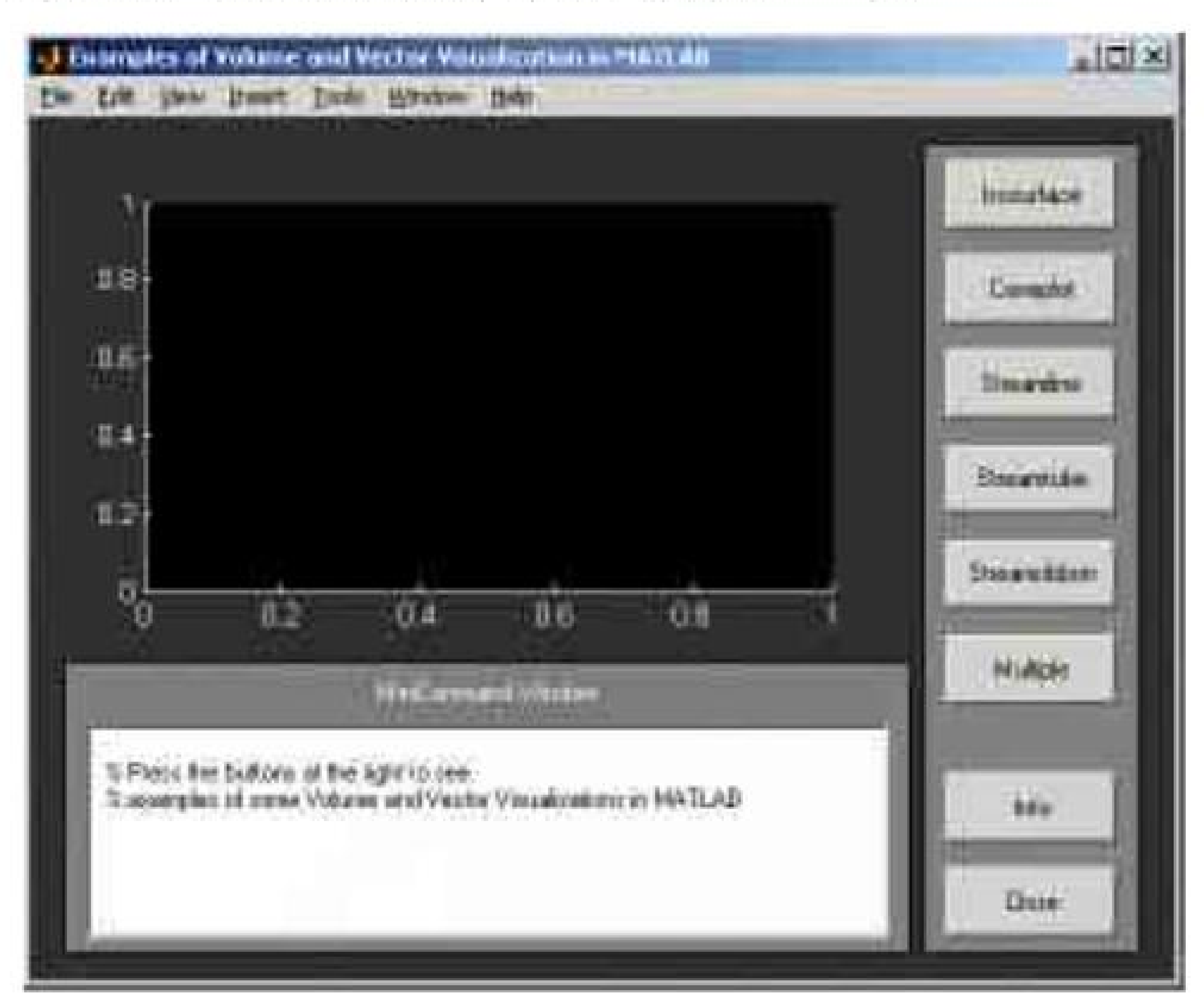


图 7-1　MATLAB 图形用户界面示例

通过如图 7-1 所示的界面，用户可以不必过多了解内部细节就可以使用 MATLAB 的强大数据可视化和计算的功能了。例如，单击用户界面右边的按钮，可以在图形窗体的绘图区域绘制各种图形，同时在文本显示区域显示具体命令行代码，如图 7-2 所示。

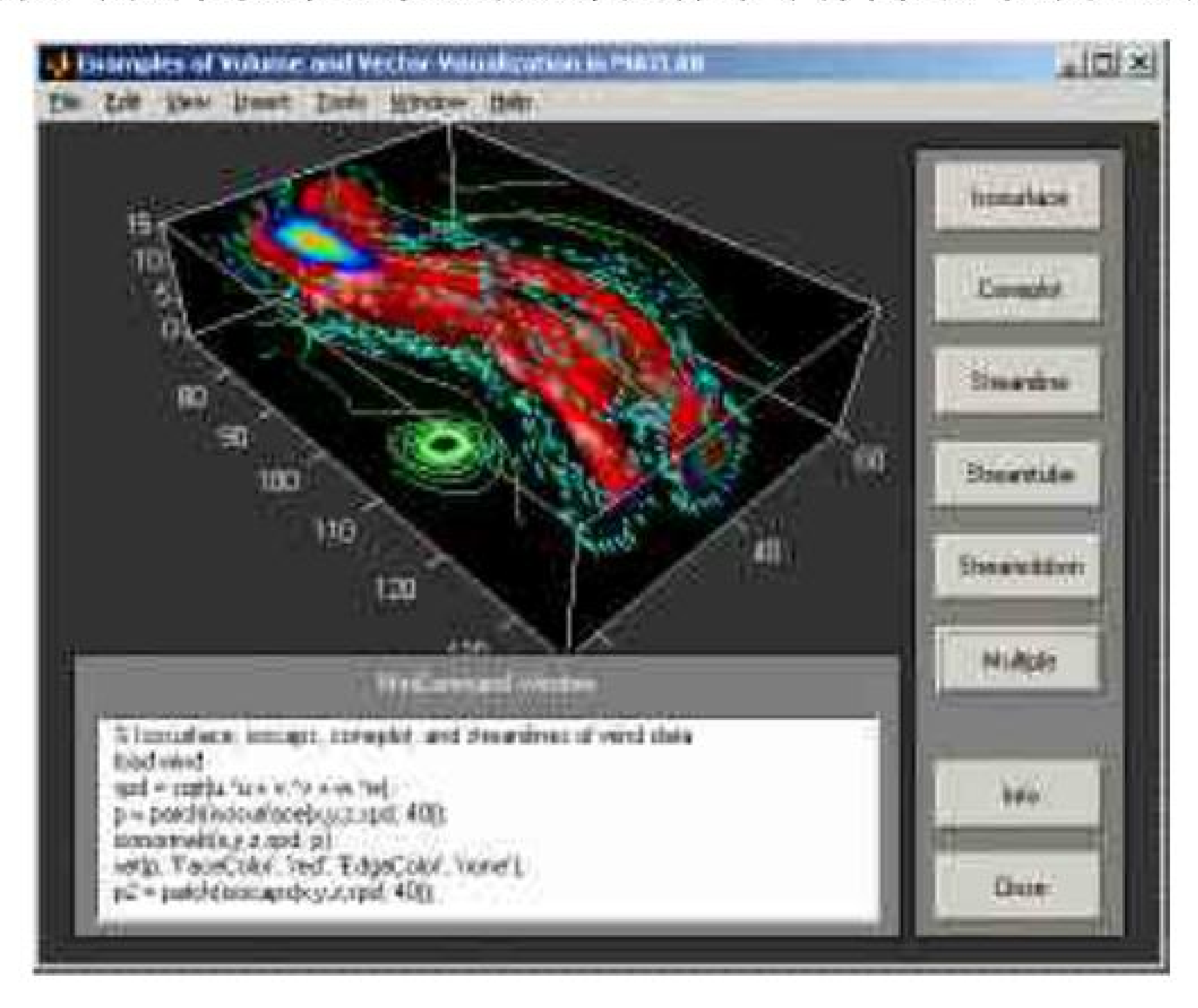


图 7-2　显示具体的算法以及图形

在本章，将结合具体的应用示例，详细介绍创建图形用户界面的基本方法——GUIDE的使用。

7.2 图形句柄入门

在第六章中介绍了很多 MATLAB 可视化函数，这些函数都是将不同的曲线或者曲面绘制在图形窗体中，而图形窗体也就是由若干图形对象组成的可视化的图形界面。在 MATLAB 环境中每一个图形对象都有一个相应的句柄，这些句柄帮助系统标识这些对象，获取或者设置它们的属性。理解图形对象句柄也是进行图形界面创建的前提之一，所以首先简要介绍图形对象句柄的概念，以及图形句柄的使用方法。

MATLAB 的图形对象是按照一定的层次排列的，如图 7-3 所示。

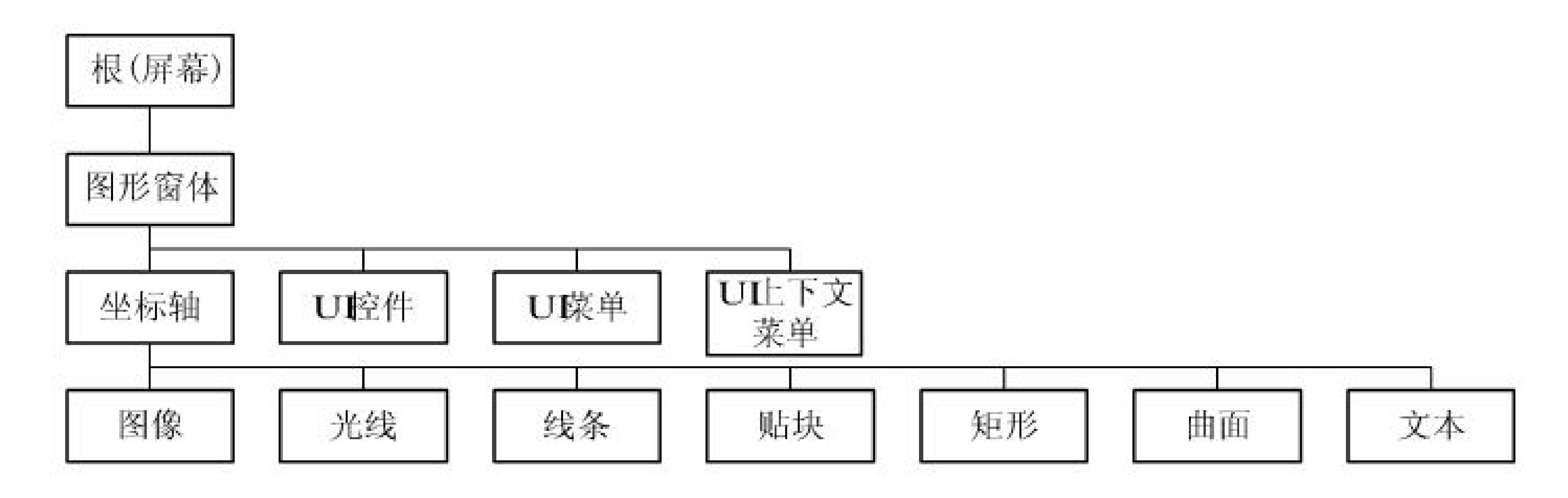


图 7-3 图形句柄的层次

在图 7-3 中，除了最上一层的屏幕(root)对象以外，每一种对象都具有自己的父层次对象，即对象的上一层次的对象，而自己下一层次的对象都被称为子对象。

具体来说，一个图形界面在 Windows 操作系统中的层次分布如图 7-4 所示。

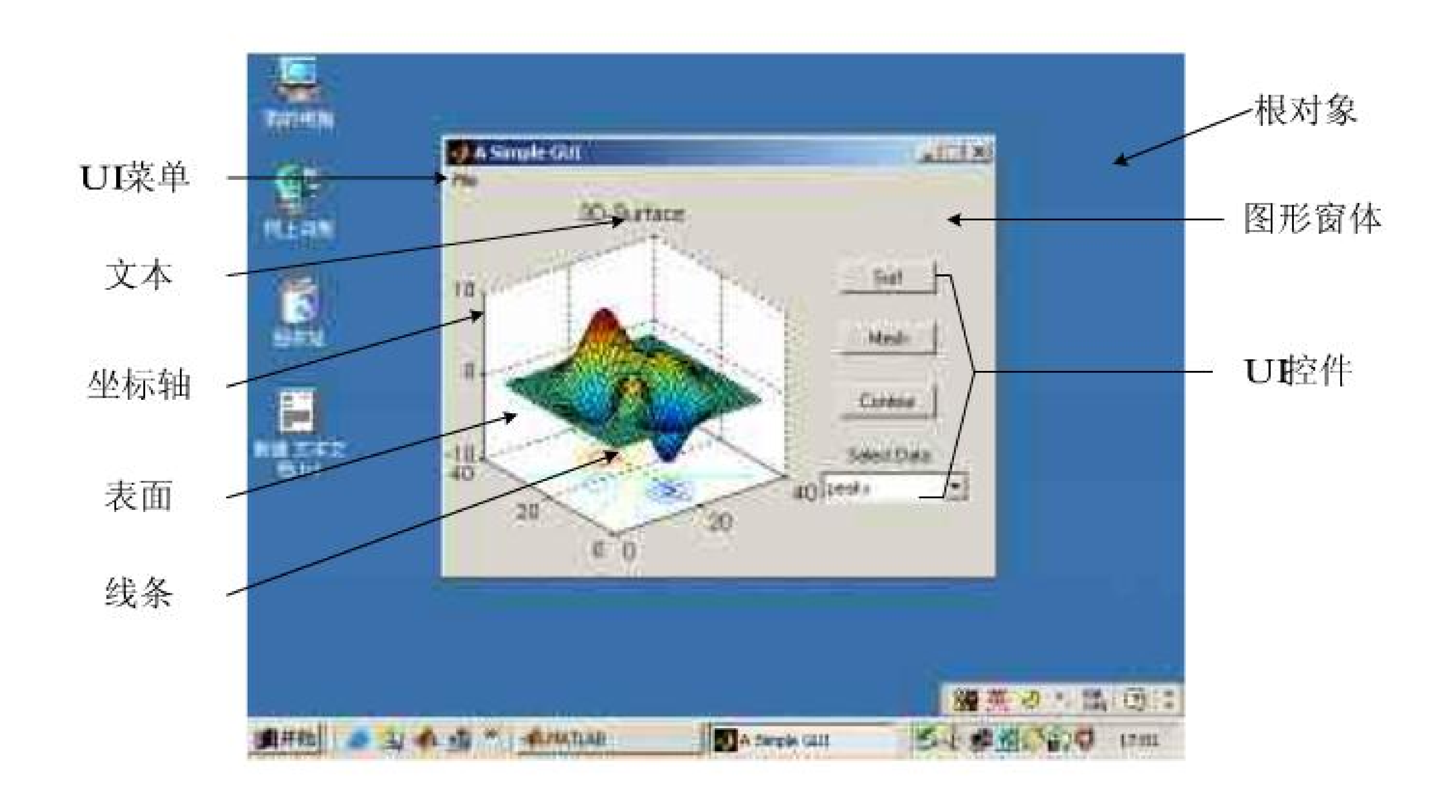


图 7-4 图形句柄的具体层次

在 MATLAB 中，只要获得了图形对象的句柄，就可以通过对属性的操作来修改图形对象的外观，这也是 MATLAB 图形用户界面编程的基本原理。MATLAB 提供了若干函数用来对图形句柄进行操作，这其中较为常用的函数在表 7-1 中进行了总结。

表 **7-1**　常用的图形句柄操作函数

函　数	说　　明
findobj	按照指定的属性来获取图形对象的句柄
gcf	获取当前的图形窗口句柄
gca	获取当前的轴对象句柄
gco	获取当前的图形对象句柄
get	获取当前的句柄属性和属性值
set	设置当前句柄的属性值

在例子 7-1 中，详细讨论了利用图形句柄修改图形对象的方法。

例子 7-1　使用图形句柄。

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> X = linspace(-pi,pi,25);
>> Y = sin(X);
>> plot(X,Y,'rX');
```

这时的图形结果为红色的以“X”为符号的正弦曲线，如图 7-5 所示。

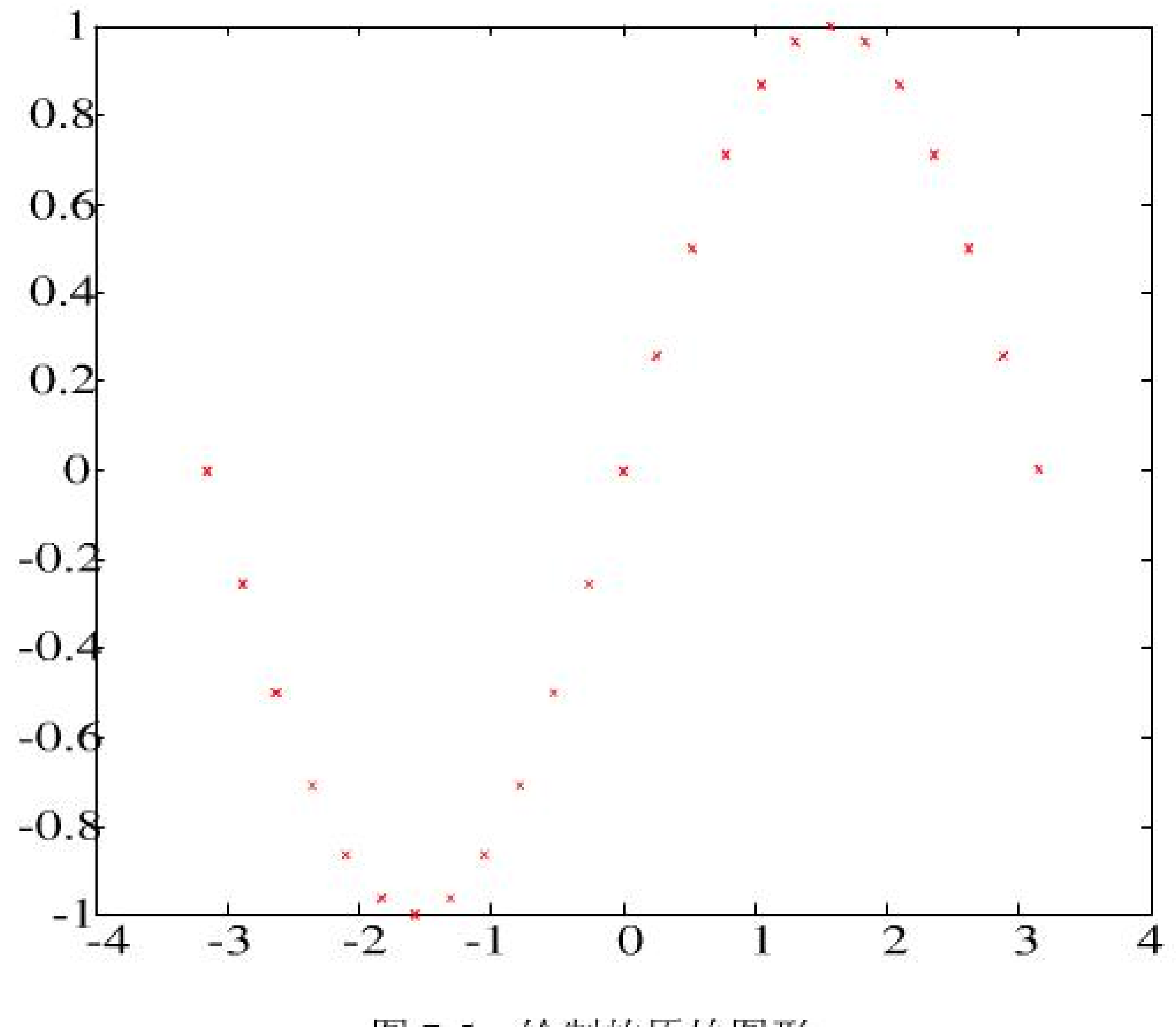


图 7-5　绘制的原始图形

获取当前的曲线对象句柄：

```
>> h_line = findobj(gca,'Marker','X')
h_line =
     3.0011
```

尽管这里 h_line 为一个双精度的数值，但这个数值变量具有特殊的意义，它代表了在当前坐标轴上绘制的曲线，可以通过这个变量来操作曲线对象，例如获取整个曲线的属性

列表：

```
>> get(h_line)
    Color = [1 0 0]
    EraseMode = normal
    LineStyle = none
    LineWidth = [0.5]
    Marker = x
    MarkerSize = [6]
    MarkerEdgeColor = auto
    MarkerFaceColor = none
    XData = [ (1 by 25) double array]
    YData = [ (1 by 25) double array]
    ZData = []

    BeingDeleted = off
    ButtonDownFcn =
    Children = []
    Clipping = on
    CreateFcn =
    DeleteFcn =
    BusyAction = queue
    HandleVisibility = on
    HitTest = on
    Interruptible = on
    Parent = [101.001]
    Selected = off
    SelectionHighlight = on
    Tag =
    Type = line
    UIContextMenu = []
    UserData = []
    Visible = on
```

这里罗列了能够获取的图形对象的属性，现在获取具体的属性：

```
>> h_line_parent = get(h_line,'Parent');
```

上述命令行获得了曲线的父对象，即当前的坐标轴：

```
>> isequal(h_line_parent,gca)
ans =
     1
```

设置曲线的属性：

```
>> set(h_line,'Color',[1 1 1],'MarkerSize',10);
```

上述的命令行将曲线设置为白色，同时将符号的大小设置为 10，不过这个时候的坐标轴也是白色，所以看不出曲线。

设置坐标轴的属性：

```
>> set(gca,'Color',[0,0,0])
```

这时坐标轴的背景色成为黑色，曲线的符号为白色，所以曲线可以被看到了。

```
>> set(gca,'XGrid','on','GridLineStyle','-.','XColor',[0.75 0.75 0])
>> set(gca,'YGrid','on','GridLineStyle','-.','YColor',[0 0.75 0.75])
```

上述的两条指令将坐标轴的网格线绘制了出来，而且使用了点划线，分别设置了不同的颜色。

```
>> set(gcf,'Color',[0 0 1])
```

这条指令将整个图形窗体的背景色设置为蓝色，这样，所有的指令综合在一起得到的效果如图 7-6 所示。

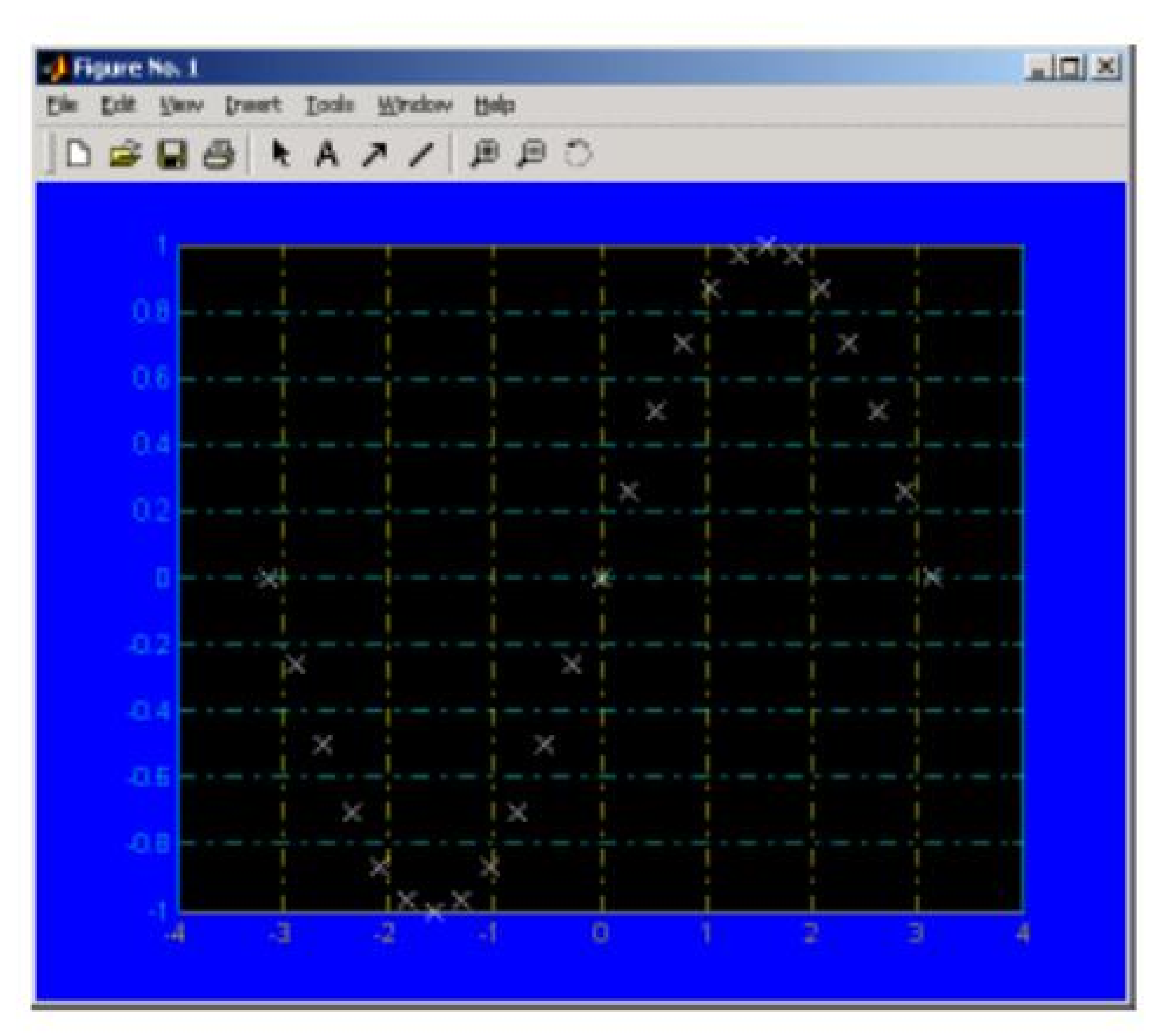


图 7-6　设置不同图形对象属性得到的结果

在 MATLAB 中，不同图形对象有不同的属性，由于受篇幅的限制，这里就不一一列举了，有兴趣的读者请参阅 MATLAB 的帮助文档。

从例子 7-1 中可以看出 MATLAB 图形界面应用的基本过程，无论是利用 GUIDE 还是图形句柄来创建图形用户界面，其基本过程都是首先获取当前的图形对象句柄，然后利用 get 函数获取一些属性——获取动作，再通过 set 函数设置一些属性——完成响应。

除了能够直接设置具体图形对象的属性以外，MATLAB 还允许用户对图形对象的默认属性进行修改。所谓图形对象的默认属性，就是那些 MATLAB 图形对象所固有的预定义的“出厂设置”，例如在默认的条件下图形窗体的背景色为深灰色，而坐标轴的背景色为白色等，在没有指定特殊的属性值之前，MATLAB 就使用这些默认的图形对象属性来显示图形对象。

若需要修改 MATLAB 的默认属性，则使用下面的命令行：

```
set(ancestor,'Default<Object><Property>',<Property_Val>)
```

其中，ancestor 为某一层次的图形对象句柄，若该句柄距离根对象越近，则影响的对象就越多，也就是说，若在根层次设置了默认属性，则所有的对象都继承这个默认属性，若在轴层次设置默认属性，则轴层次以下的对象继承该默认属性。下面举例说明设置对象默认属性的方法。

例子 7-2　设置修改对象的默认属性。

本例子使用的脚本文件内容如下：

```
% 修改图形窗体默认背景色
set(0,'DefaultFigureColor',[1 1 1]);
% 修改默认的坐标轴背景色
set(0,'DefaultAxesColor',[0 0 0]);
% 修改坐标线的色彩
set(0,'DefaultAxesXColor',[0.5 0 0]);
set(0,'DefaultAxesYColor',[0.5 0 0]);
% 修改文本的色彩
set(0,'DefaultTextColor',[0 0.5 0]);
X = linspace(-pi,pi,25);
Y = sin(X);
plot(X,Y,'yX');
grid on
title('Change The Default Properties');
legend('sin');
```

运行例子 7-2 的脚本文件，将修改部分对象的默认属性值，所以得到的图形结果如图 7-7 所示。

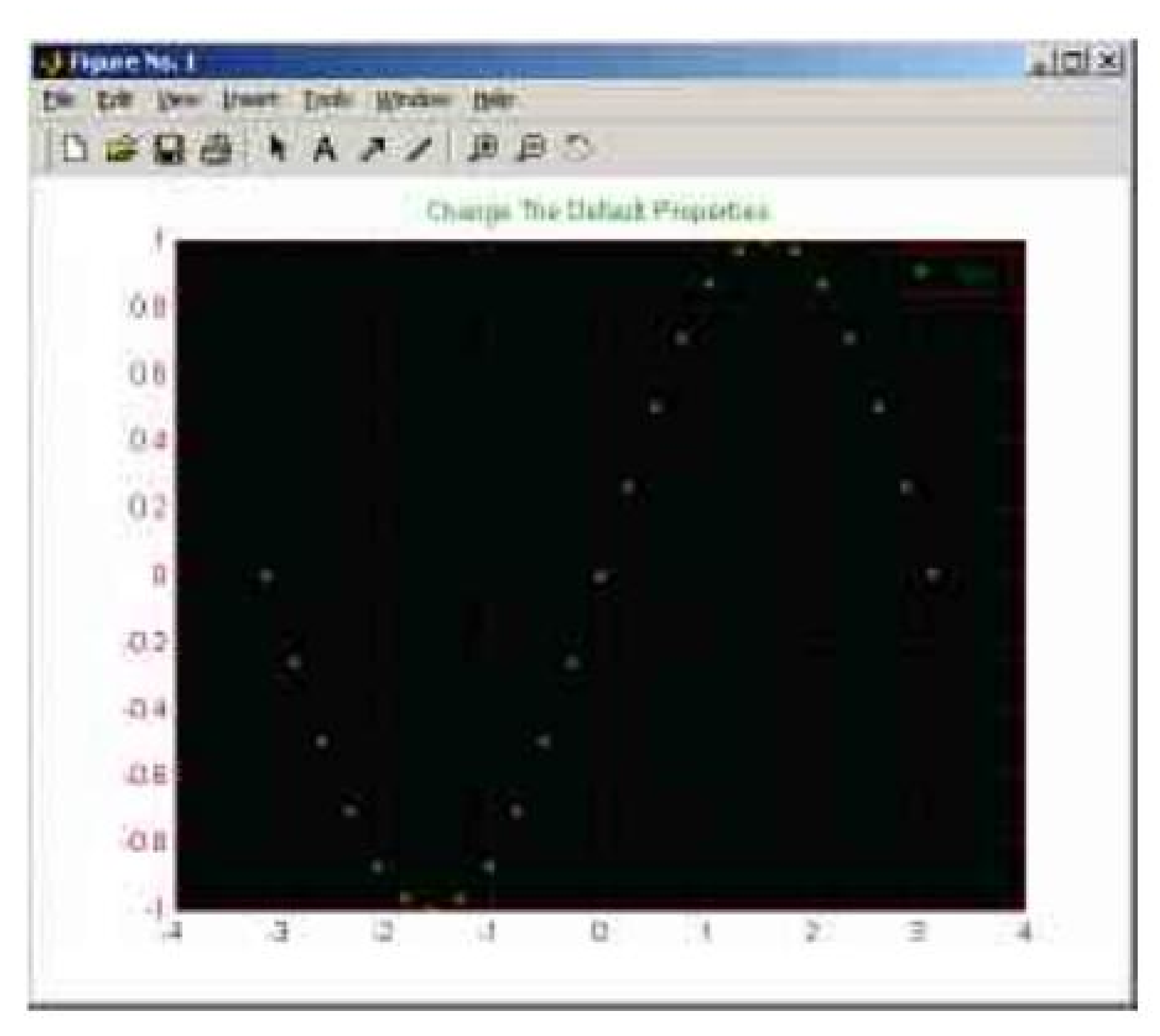


图 7-7　修改默认属性值后的显示效果

修改的默认属性在本次 MATLAB 会话期间都有效，若关闭了 MATLAB 再次启动之后，则这些默认的属性就会恢复“出厂设置”，所以，若希望设置的默认属性在每次启动 MATLAB 时都发挥作用，则需要在 startup.m 文件中添加修改默认设置的指令。注意例子 7-2 的代码，这里首先修改了默认属性值，然后再来进行图形的绘制。

提示：

如果希望将已经修改的默认属性值恢复成出厂设置，则可以使用下面的命令行：

```
>> set(h, 'PropertyName', 'default')
```

或者

```
>> set(h, 'PropertyName', 'factory')
>> set(h, 'PropertyName', 'remove')
```

7.3　GUIDE 工具入门

在 MATLAB 中创建图形用户界面有两种方法，其中之一是使用图形句柄，用这种方法创建图形界面的过程相当繁琐，而且在程序编写好之前，用户图形界面是不可见的。所以为了便于创建图形用户界面，MATLAB 提供了一个开发环境，能够帮助用户创建图形用户界面，这就是 GUIDE——Graphic User Interface Development Environment。在本小节将介绍 GUIDE 的基本使用方法。

在 MATLAB 中启动 GUIDE 的方法是在 MATLAB 命令行中键入指令：

```
>>guide
```

或者通过“Start”菜单选择“MATLAB”下的“GUIDE”命令。

这时在 MATLAB 6.5 中，将直接启动 GUIDE Quick Start 窗体，在这个窗体中，可以初步选择图形用户界面的类型，如图 7-8 所示。

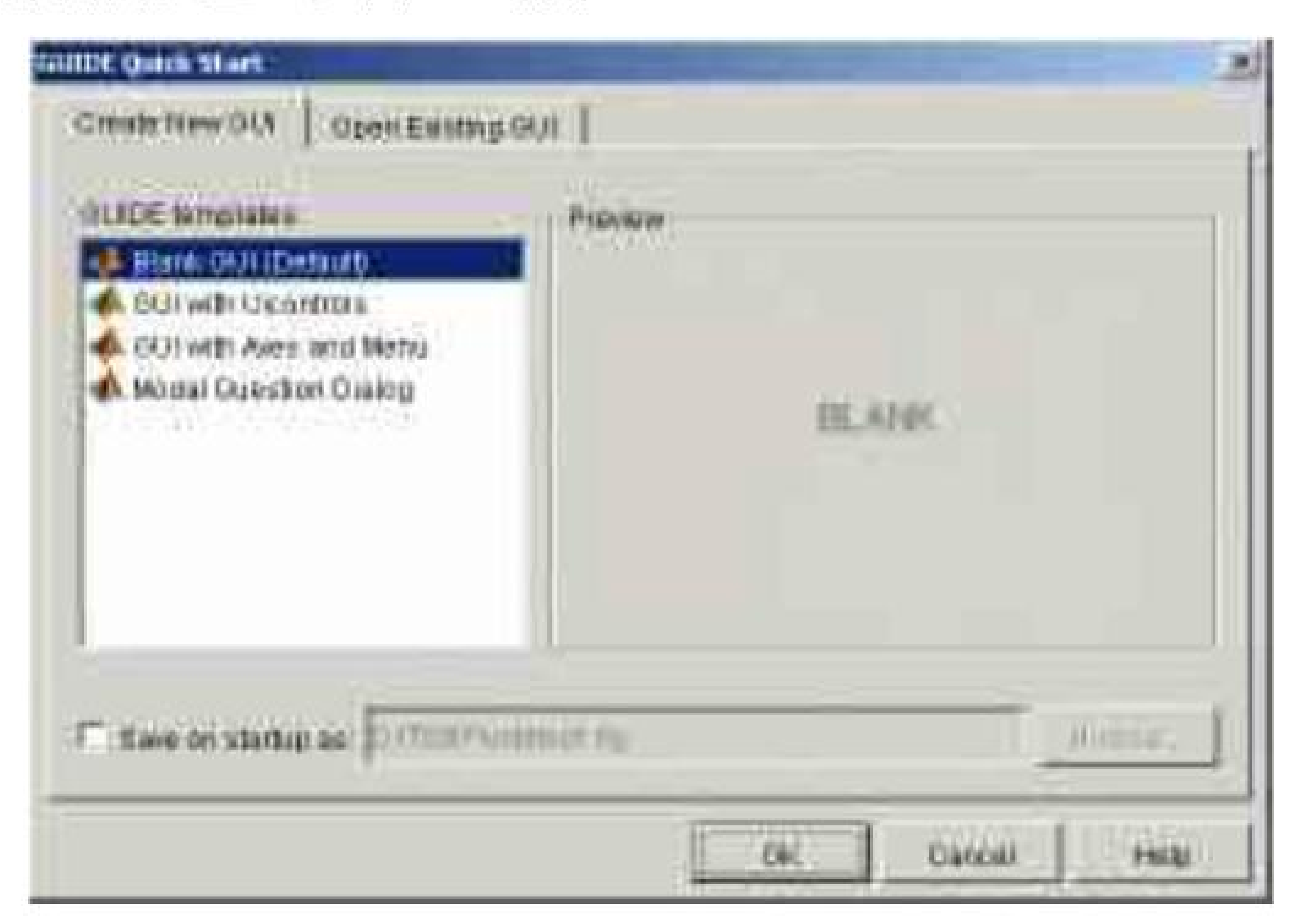


图 7-8　GUIDE 的快速启动界面

在快速启动界面中，可以选择四种类型的新建界面：

■ 空白界面(Black GUI)。

■ 具有图形控件的界面(GUI With Uicontrols)。
■ 具有菜单和坐标轴的界面(GUI With Axes and Menu)。
■ 模式对话框(Modal Question Dialog)。

其中，空白界面同早期版本的 GUIDE 图形用户界面一致。用户可以根据自己的需要选择不同的初始界面类型，以加快自己的开发任务。在本书中，仅讨论通过空白界面创建图形用户界面的方法，而其他的界面类型请参阅 MATLAB 的帮助文档。

除了能够创建新建的图形界面之外，还可以选择已经存在的图形界面文件，该文件的扩展名为 fig，是 MATLAB 自己的图形文件格式。也可以通过下面的命令行直接打开一个存在的 GUI 界面文件：

```
>> guide gui_filename
```

这时在 GUIDE 中将显示已经创建好的图形界面外观。

在本章中，选择空白界面类型，并单击“OK”按钮，这时 MATLAB 将启动 GUIDE 的图形界面，如图 7-9 所示。

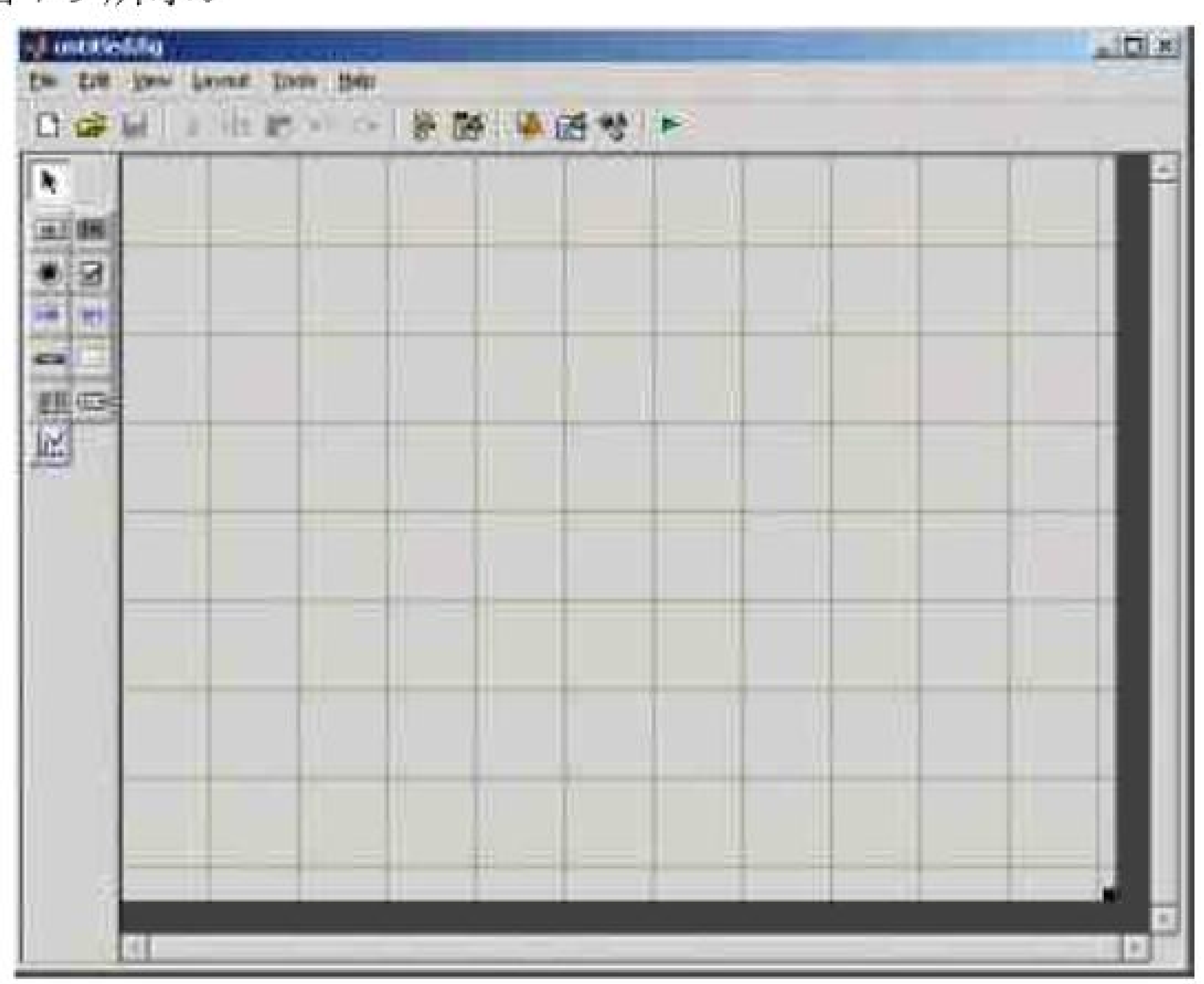


图 7-9 具有空白界面的 GUIDE 图形界面

在 GUIDE 界面中，位于中央的深灰色部分为绘制控件的画布，用户可以调整画布的尺寸以得到不同的界面尺寸。

在 GUIDE 界面的左侧为 MATLAB 的控件面板，控件面板包含了能够在画布上绘制的图形控件，其中分别为推按钮(Push Button)、单选按钮(Toggle Button)、单选框(Radio Button)、复选框(Checkbox)、文本框(Edit Text)、静态文本框(Static Text)、滚动条(Slider)、组别框(Frame)、列表框(Listbox)、下拉框(Popup Menu)和坐标轴(Axes)。创建 GUIDE 的过程之一，就是从控件面板中选择需要的控件，然后用鼠标在画布上绘制合适大小的控件。控件面板的外观可以通过设置 GUIDE 的属性进行简要的修改，选择 GUIDE 中“File”菜单下的“Reference”命令，在弹出的对话框中选择“Show names in Component palette”复选框，如图 7-10 所示。

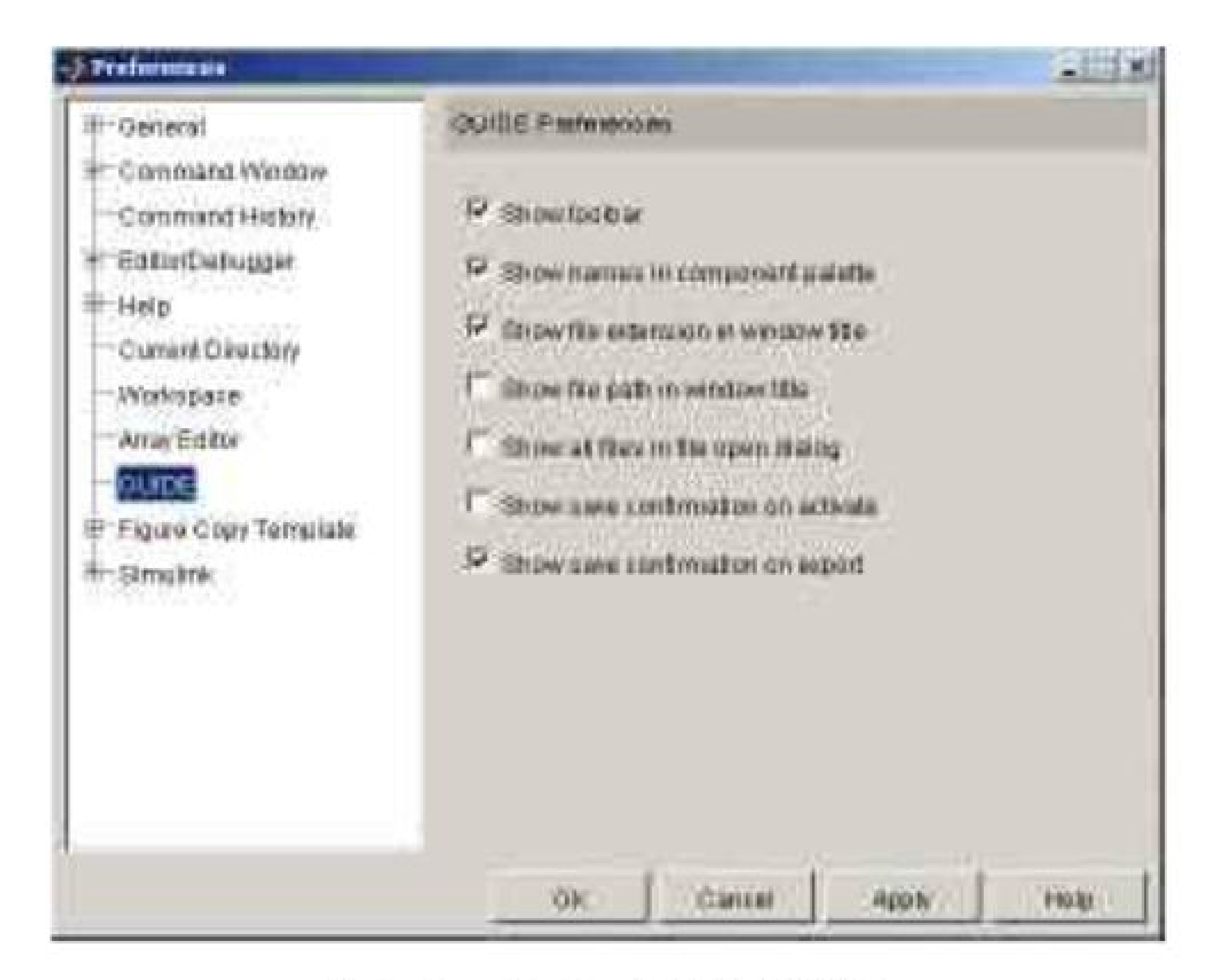


图 7-10　GUIDE 的属性对话框

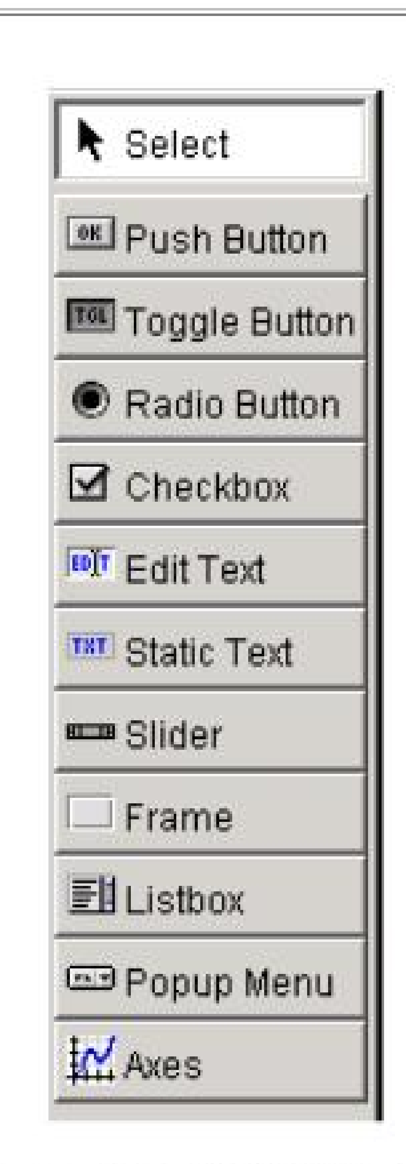


图 7-11　显示不同控件的名称

单击“OK”按钮之后，控件面板中在不同的控件旁边会显示相应控件的名称，如图 7-11 所示。

大家可以根据自己的喜好选择控件栏上的显示样式。

此外，在 GUIDE 的界面的工具栏上包含了一些常用的工具，如图 7-12 所示。

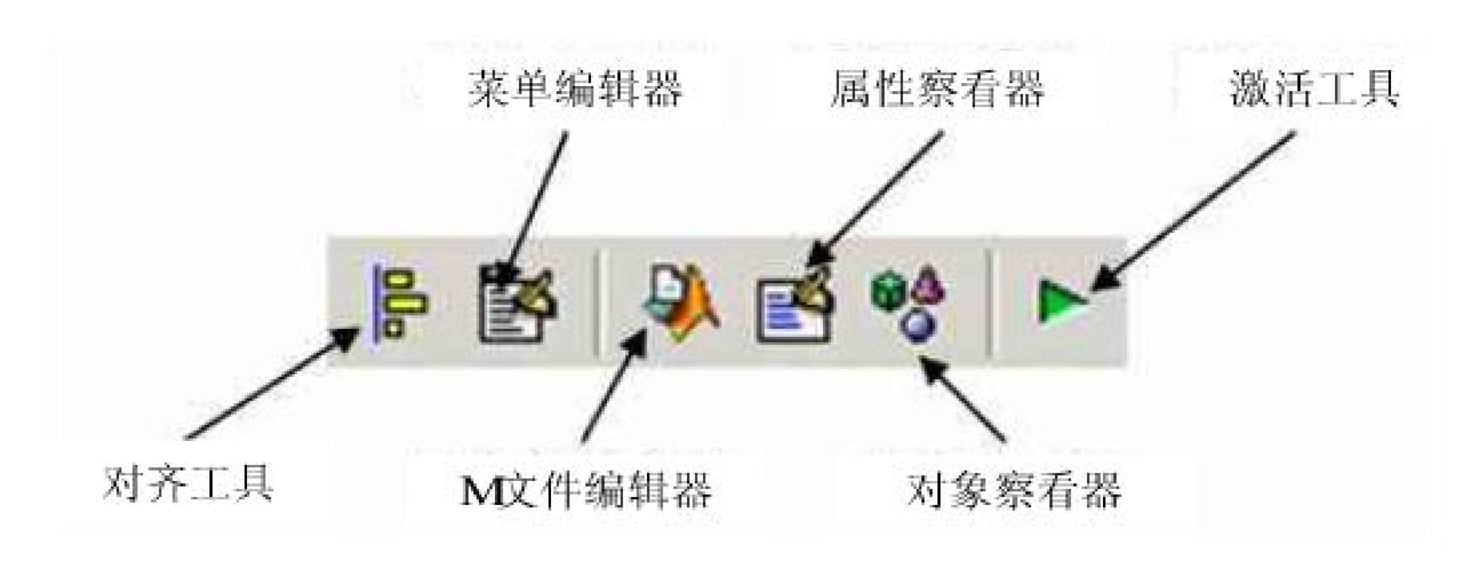


图 7-12　GUIDE 工具栏上的工具按钮

这些常用的工具可以帮助用户提高创建图形用户界面应用程序的效率，具体工具的使用方法将在本章中逐步介绍。

利用在本小节介绍的 GUIDE 工具就可以来创建图形用户界面了。在后面的两个小节中，将结合具体的实例来介绍创建图形用户界面的一般方法和过程，在这里首先给出图形用户界面的结果，如图 7-13 所示。

在该图形用户界面中包含如下控件：

- ■ 两个推按钮(push button)，分别完成绘制三维曲面和改变色彩的功能；
- ■ 五个静态文本框(static text)，分别用来完成显示不同信息的功能；
- ■ 一个滚动条(slide)，用来完成改变三维曲面上的分隔线色彩；
- ■ 一个坐标轴(axes)，用来显示三维曲面；
- ■ 一个菜单(menu)，用来完成清除坐标轴的功能。

在后面两个小节中，将详细介绍创建该图形用户界面的方法和步骤。

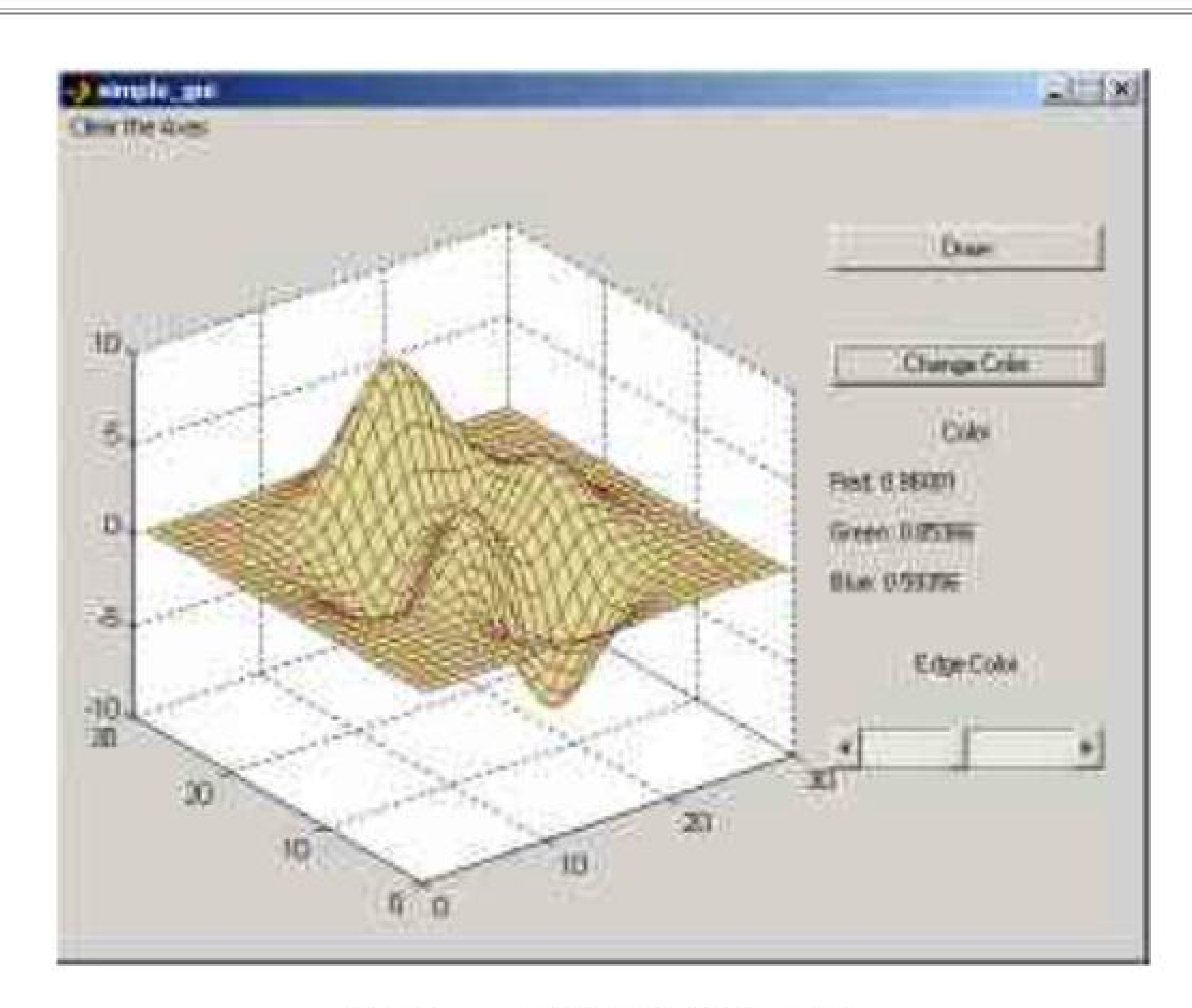


图 7-13 图形用户界面的例子

7.4 创建图形用户界面外观

在本章需要实现的图形用户界面中，需要包含一个坐标轴用来显示三维表面(surface)，而且界面中还需要包含两个按钮，分别用来在坐标轴中绘制三维表面和修改三维表面的颜色。在修改颜色时，需要通过几个文本框将颜色数值显示出来。通过滚动条可以修改三维表面的网格线色彩。最后，在图形界面上还有一个菜单，通过菜单命令可以清除当前坐标轴的内容。那么创建这样一个图形用户界面大体的步骤是怎样的呢？

首先，进行界面设计。在这一过程中，需要对界面空间的布局、控件的大小等进行设计，最好的方法就是在一张纸上简要地绘制一下界面的外观，做到心中有数。

然后，利用 GUIDE 的外观编辑功能，将必要的控件依次绘制在界面的“画布”上。在这一过程中，主要将所有控件摆放在合适的位置，并且设置控件合适的大小。

第三步，设置控件的属性，这一步骤重点需要设置控件重要的属性值，例如控件的回调函数、标签和显示的文本等。

第四步，也就是最后一步，就是针对不同的控件需要完成的功能进行 M 语言编程。

在本小节将详细介绍创建图形用户界面外观的方法，而设置控件属性和实现不同控件功能的 M 语言编程，将在下一小节讲述。

在进行了简单界面设计之后，就要创建具体的图形用户界面外观了。首先通过 GUIDE 的快速启动界面打开一个具有空白窗体的 GUIDE 界面。然后选择界面中“Tools”菜单下的“GUI Options”命令，打开 GUI 选项设置对话框。在这个对话框中，将首先设置图形界面的一些行为，例如，在创建用户界面外观的阶段，选择“Generate Fig File Only”单选框，这样，创建的用户界面仅创建外观图形文件，而不会创建相应的 M 文件，便于在创建图形用户界面外观的阶段检查创建的效果，如图 7-14 所示。



图 7-14　设置界面应用程序的属性

单击“OK”按钮之后再次选择“Tools”菜单下的“Grid and Rulers”命令，在弹出的对话框中可以设置画布上网格线的尺寸，画布上的网格线可以帮助用户来设置控件的尺寸以及确定对齐控件的位置，所以需要选择合适的网格尺寸，默认的数值为 50 像素，可以根据需要设置自己的数值，在本例子中，设置尺寸为 20 个像素，如图 7-15 所示。

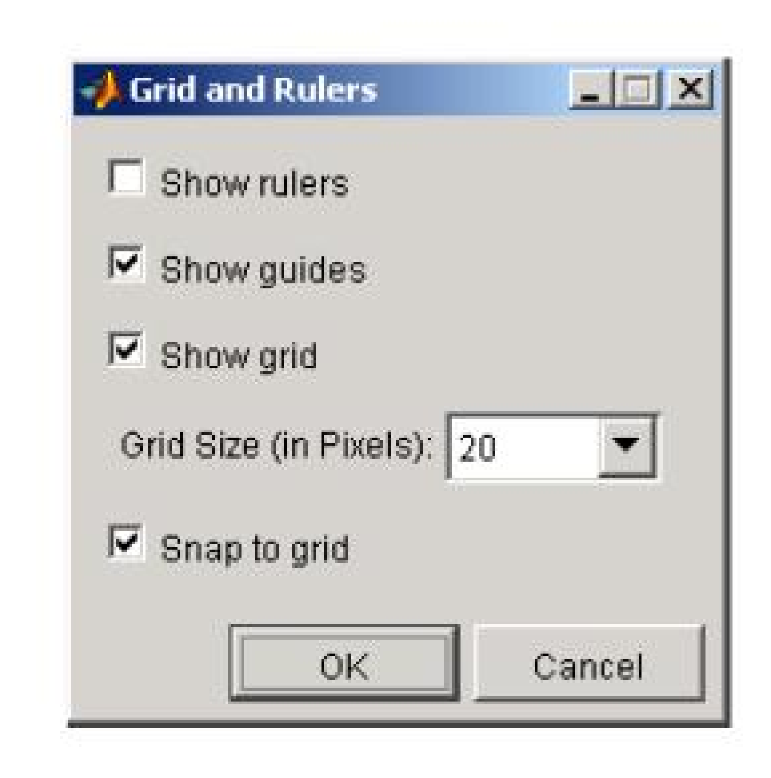


图 7-15　设置网格尺寸

对开发环境进行了简要设置之后，接下来就要设置画布的尺寸，画布的尺寸也就是图形界面的未来尺寸，需要在绘制控件之前将界面的尺寸大体确定下来。

之后，就要将不同的控件绘制在画布上了，这个过程相对来说比较简单，而且绘制控件的时候暂时可以不用考虑控件的尺寸和位置对齐等问题，只要大致将不同的控件放置在相应的位置上即可，如图 7-16 所示。

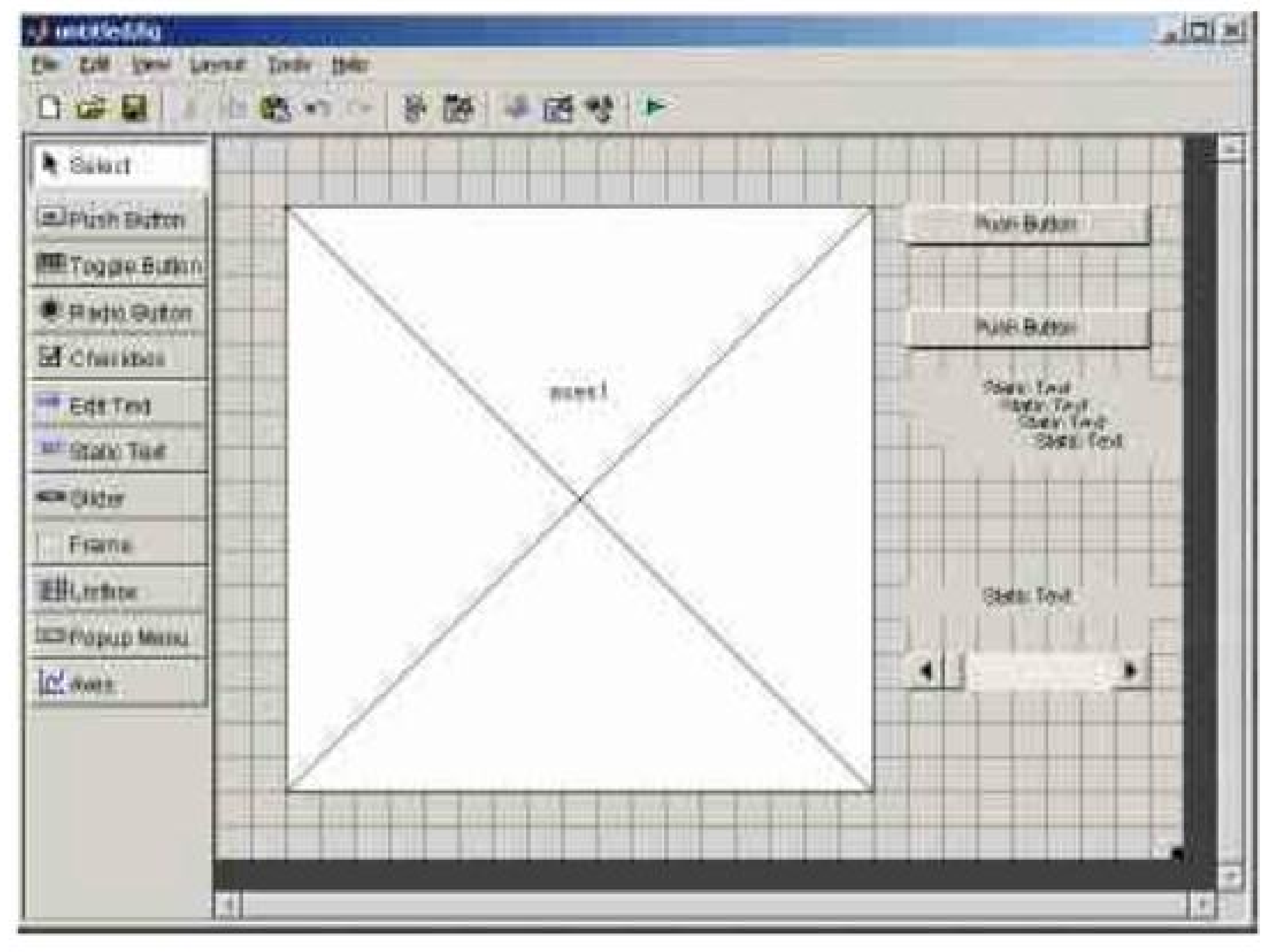


图 7-16　放置图形用户界面的控件

这时的图形界面中包含了必要的图形界面控件，不过用来显示信息用的静态文本框没有排列好，显得非常凌乱，这个时候可以使用 GUIDE 的排列工具完成控件的排列工作。这里需要排列的控件是四个静态文本框，首先将最后一个静态文本框放置在理想的位置上，如图 7-17 所示。

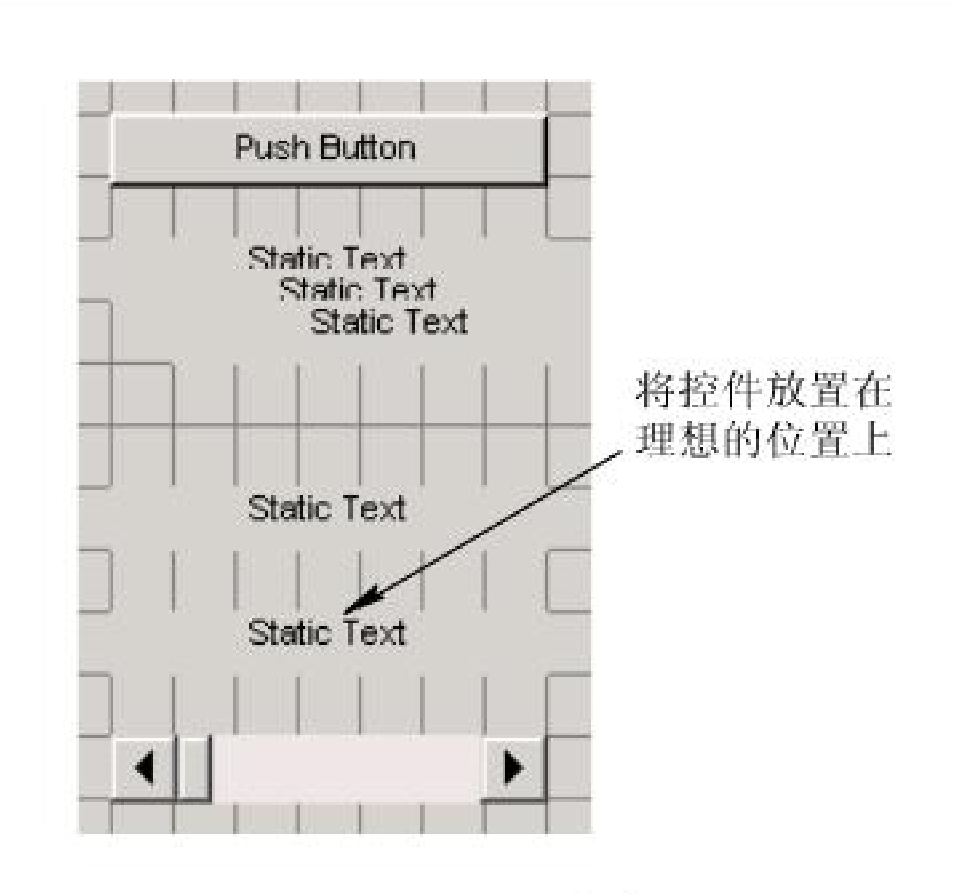


图 7-17 设置必要控件的位置

然后选择四个静态文本框，单击 GUIDE 工具栏中的对齐工具按钮，在弹出的排列工具对话框中，分别选择垂直方向上均匀分布、水平方向上左边界对齐按钮，如图 7-18 所示。

单击“Apply”按钮之后，可以察看对齐控件之后的效果，如图 7-19 所示。



图 7-18 对齐工具对话框

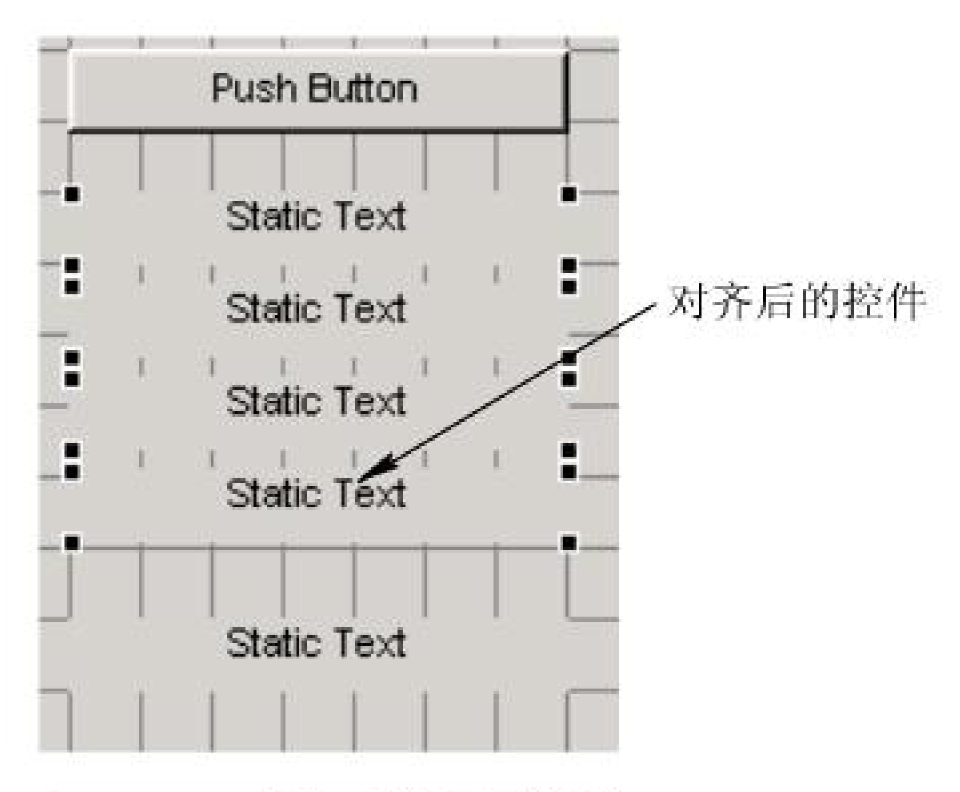


图 7-19 对齐后的界面控件

在界面之中还需要创建菜单，创建菜单可以通过菜单编辑器完成。单击工具栏上的菜单编辑器按钮，可以打开菜单编辑器对话框，在对话框中单击创建新菜单按钮，则可以创建新的菜单，设置菜单属性如图 7-20 所示。

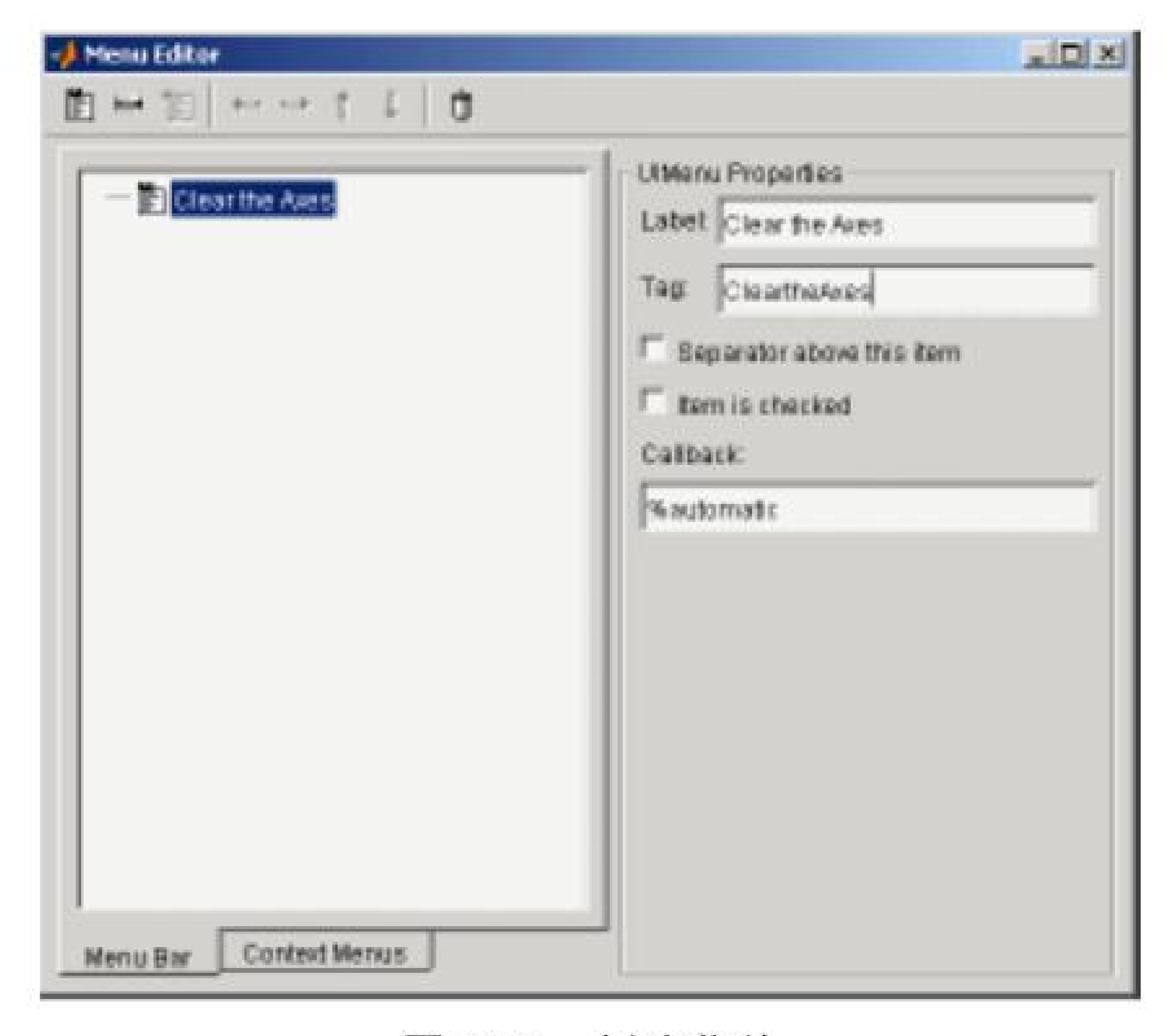


图 7-20 创建菜单

注意

这里在创建菜单时，同时需要设置菜单的 Label 属性和 Tag 属性，Tag 属性将在后面编写界面应用程序时使用。为了能够让菜单发挥作用，还需要添加一个菜单项，单击新建菜单项按钮，同样在菜单编辑器对话框中设置菜单项的 Label 属性和 Tag 属性分别为 Done 和 ClearAxesDone。

到现在，整个图形界面元素就基本上创建完毕了，这时可以单击 GUIDE 工具栏中的 Run 按钮，激活图形界面，由于在前面的步骤中，设置了仅生成 Fig 文件，所以这时可以利用激活界面的方法来考察界面的布局状况，如图 7-21 所示。

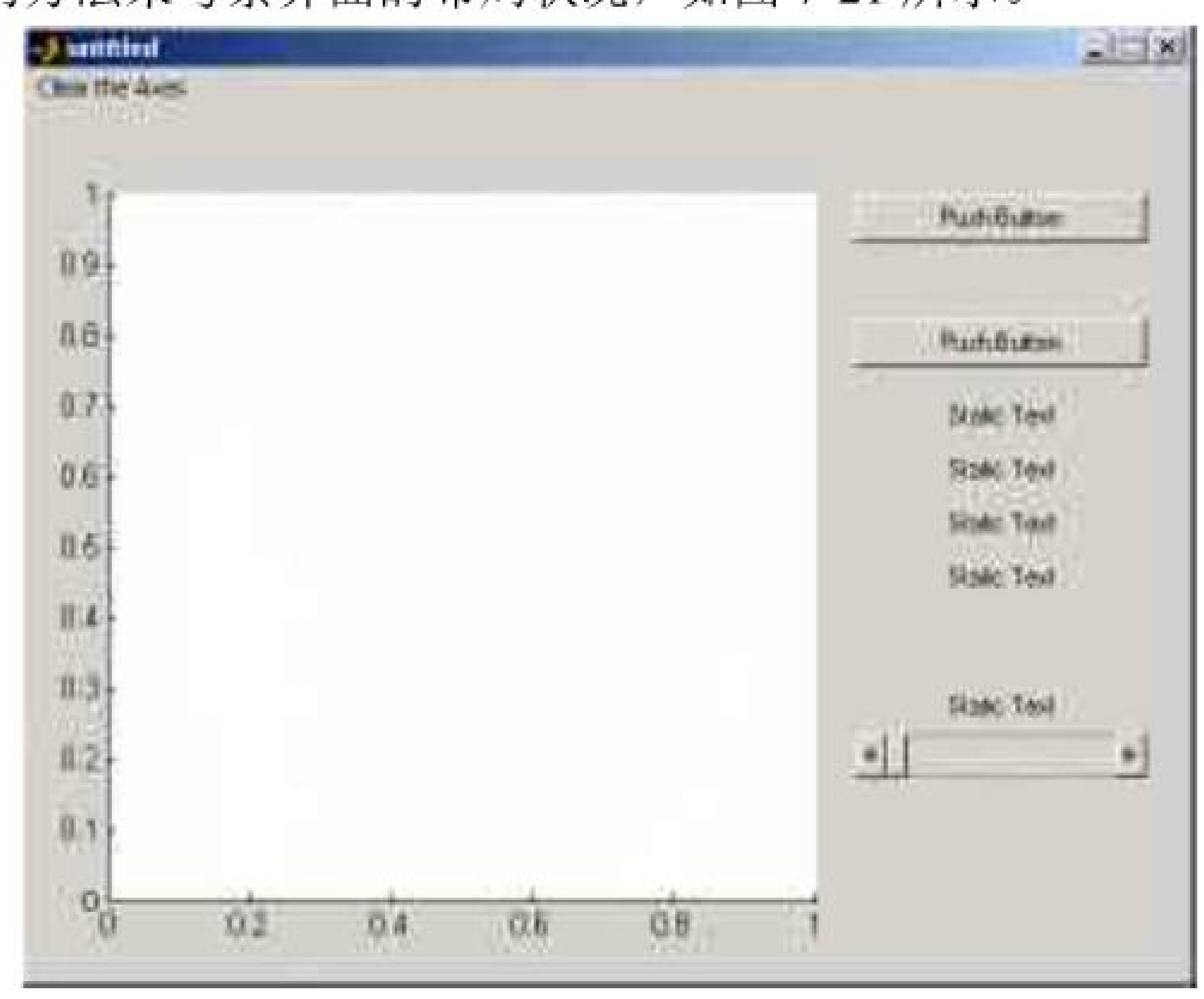


图 7-21　激活界面察看效果

到现在为止，图形用户界面外观的创建就告一段落了，如果用户对创建的图形界面不满意，还需要进一步地调整和修改，否则，就进入图形用户界面编程的步骤。

7.5　图形用户界面编程

尽管在前面的小节已经得到了图形用户界面，但是现在的图形界面还不能实现任何功能，它不能响应用户的输入，也不能在界面的坐标轴中绘制图形对象，这些功能需要通过编写 M 语言应用程序完成。进行图形用户界面编程的工作主要有两个步骤，首先设置控件的属性，然后再针对不同的控件进行 M 语言编程。

7.5.1　设置对象属性

MATLAB 的图形对象都有不同的属性，在所有的属性中，比较重要的是控件的 String 属性和 Tag 属性，前者为显示在控件上的文本，后者相当于为控件取个名字，这个名字为控件在应用程序中的 ID，控件的句柄和相应的回调函数都与这个名字有直接的关系。设置控件的属性可以使用 GUIDE 的属性察看器和控件浏览器完成。

单击工具条中的控件浏览器按钮，在弹出的对话框中，可以察看所有已经添加在图形界面中的对象以及对象的 String 和 Tag 属性，如图 7-22 所示。

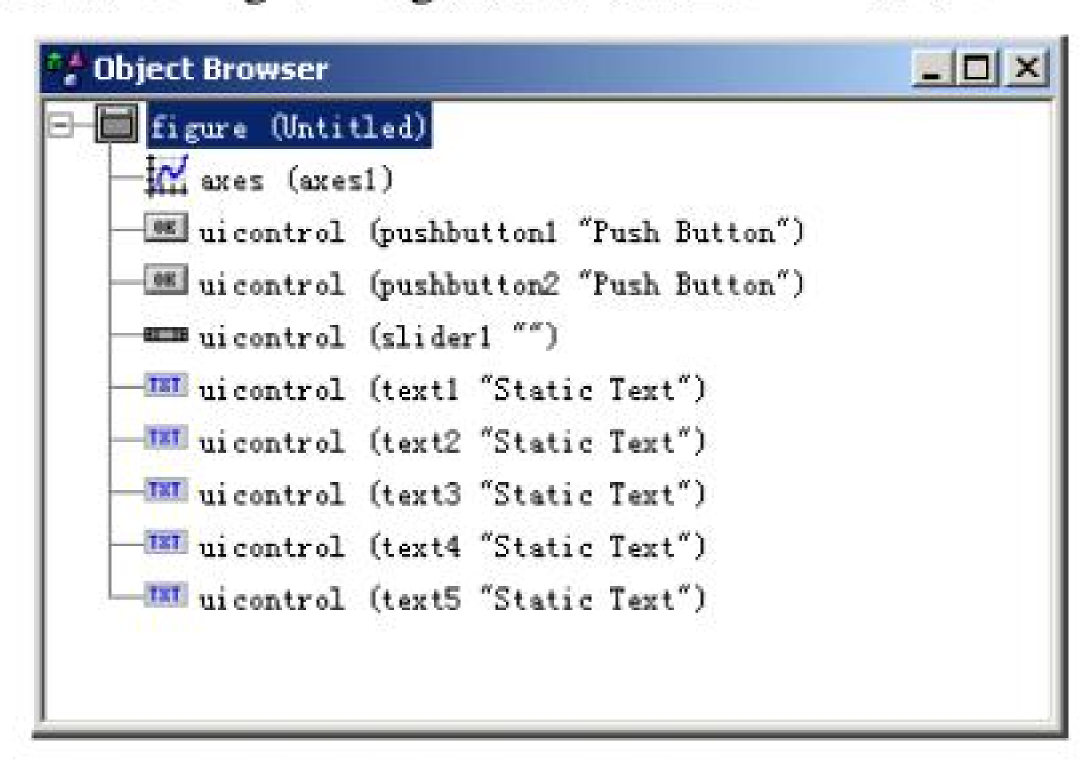


图 7-22 控件对象浏览器

首先设置图形窗体的属性，用鼠标双击控件对象浏览器中的“figure(Untitled)”，可以打开属性察看器编辑修改和察看图形窗体的属性。这里需要修改的属性包括图形的 Name 属性和 Tag 属性，将 Name 属性设置为 Simple GUI，将 Tag 属性设置为 simpleGui，如图 7-23 所示。

然后双击控件对象浏览器中的 uicontrol(pushbutton1"Push Button")，这时将打开按钮对象的属性察看器，同时，在 GUIDE 的外观编辑器中，可以看到画布上的第一个按钮被选中了。这时，需要将该按钮的 String 属性设置为 Draw，将 Tag 属性设置为 btnDraw，如图 7-24 所示。

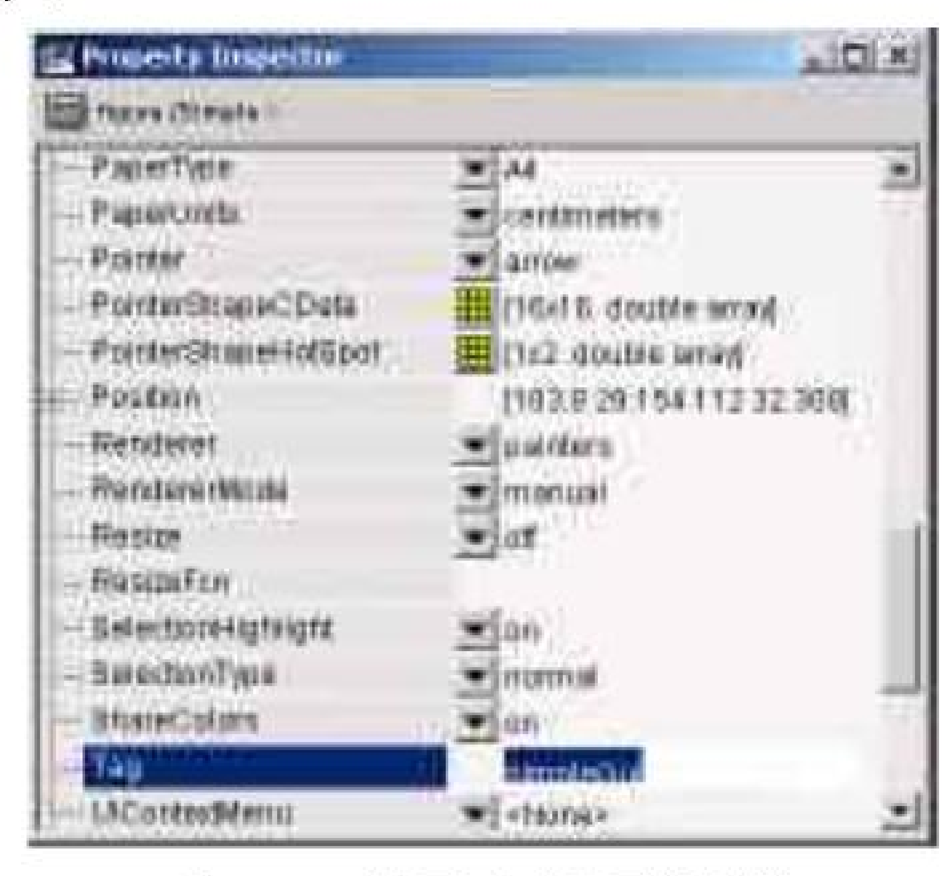


图 7-23 设置图形界面的属性



图 7-24 设置按钮的属性

依此类推，分别将其他的控件设置为如下的属性：

① 第二个按钮：

■ String : Change Color。

■ Tag : btnChangeColor。

② 静态文本框 1：

■ String : Color。

③ 静态文本框 2：

■ String : Red。

■ Tag : txtRed。
■ HorizontalAlignment : left。
④ 静态文本框 3：
■ String : Green。
■ Tag : txtGreen。
■ HorizontalAlignment : left。
⑤ 静态文本框 4：
■ String : Blue。
■ Tag : txtBlue。
■ HorizontalAlignment : left。
⑥ 静态文本框 5：
■ String : Edge Color。
⑦ 滚动条：
■ Tag : sliderEdgeColor。

注意：

在设置图形界面对象的 Tag 属性时，建议按照如下的格式进行设置：objectstyleObject-Function，即使用表示对象类型的字符串作为 Tag 属性的前缀，这样在编写控件回调函数时，能够直接从控件的名称上判断控件的类型，便于程序的管理和维护。

若此时再次激活图形界面，则得到的图形界面效果如图 7-25 所示。

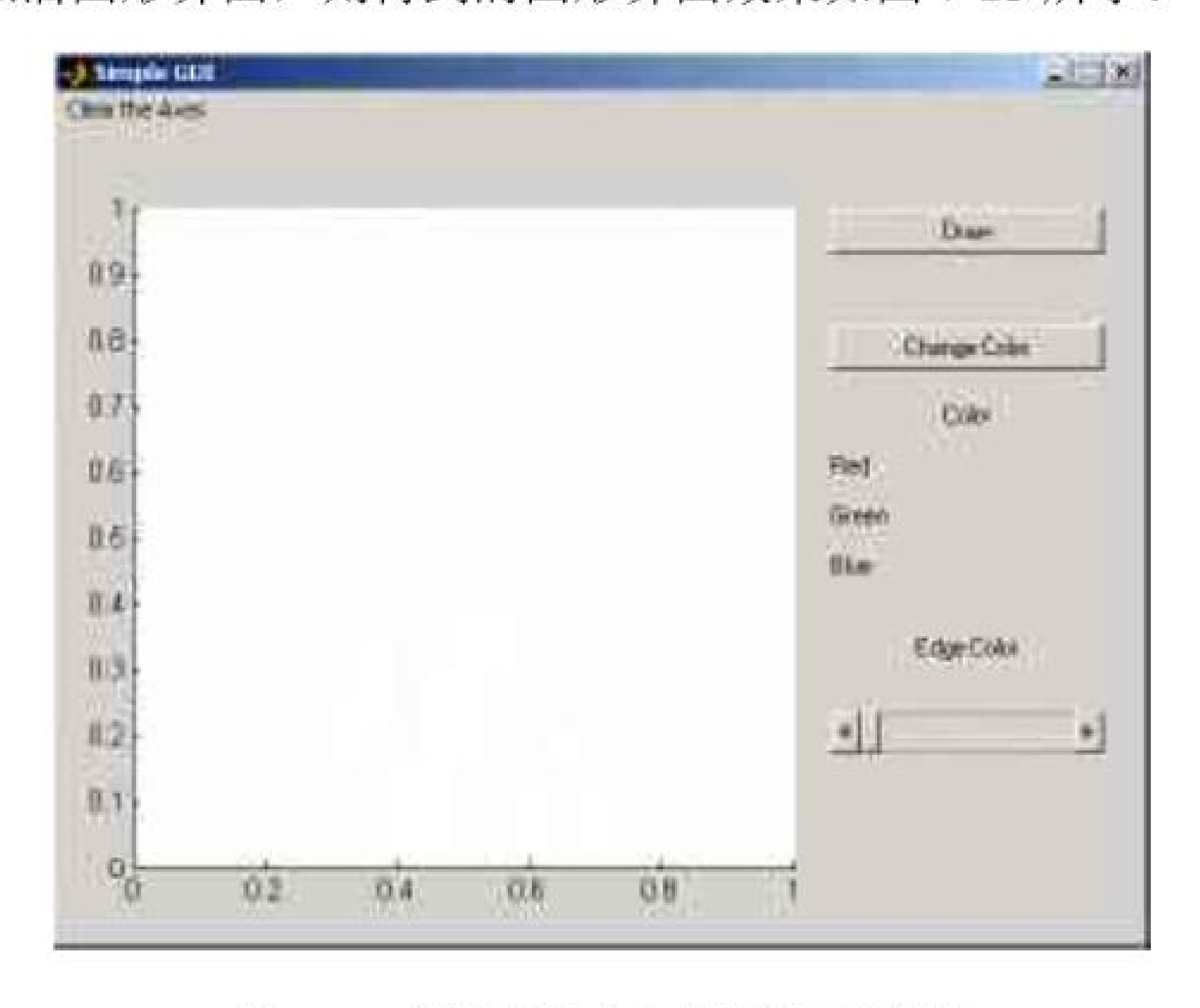


图 7-25　设置控制对象属性得到的效果

7.5.2　编写回调函数

完成了前面的工作之后，就要通过编写控件的回调函数来实现不同控件的界面功能了。图形用户界面的功能主要通过控件响应用户的动作来完成，特别在 MATLAB 的图形用户界面应用程序中，用户界面控件主要响应用户的鼠标动作——单击动作也就是选中控件的动

作。响应鼠标动作的方法通过编写回调函数来完成。

MATLAB 图形用户界面控件的回调函数，是指在界面控件被选中的时候，响应动作的 M 语言函数。在回调函数中，一般需要完成如下功能：

- 获取发出动作的对象句柄。
- 根据发出的动作，设置影响的对象属性。

例如，在单击“Draw”按钮之后，首先需要在回调函数中获取发生对象句柄，一般这一步骤都是由 MATLAB 背后的工作机制来完成的，然后设置相应对象的属性——在坐标轴上绘制相应的曲线或者图形对象，这一步骤需要用户编写具体的代码来实现。不过，利用 GUIDE 进行图形界面编程的好处之一就是，GUIDE 会为用户创建一个 M 回调函数文件的构架，一般地来说，不需要用户自己来获取发生事件的控件对象，构架文件将自动处理并将相应的句柄传递到函数中。

一般地，回调函数的声明为

```
function object_Callback(hObject , eventData , handles)
```

其中：

- object 为发生事件的控件的 Tag 属性字符串。
- hObject 为发生事件的控件的句柄。
- eventData 为保留字段，目前版本的 MATLAB 还暂时不使用。
- handles 为一个结构，这个结构中包含所有界面上控件的 Tag 属性值，还可以添加用户自己的数据。

handles 结构在图形用户界面编程中是相对比较重要的元素。主要的原因是该结构中不仅包含了所有界面对象的 Tag 属性，通过 Tag 属性可以直接像操作控件句柄一样来获取或者修改控件的属性，而且在 handles 结构中还可以添加用户自定义的需要在不同的回调函数之间共享的数据。

为了能够创建 M 构架文件，还需要执行“Tools”菜单下的“GUI Option”命令，在弹出的对话框中，选择“Generate FIG-file and M-file”单选框，一般在激活界面的同时，执行相应的 M 函数文件，如图 7-26 所示。

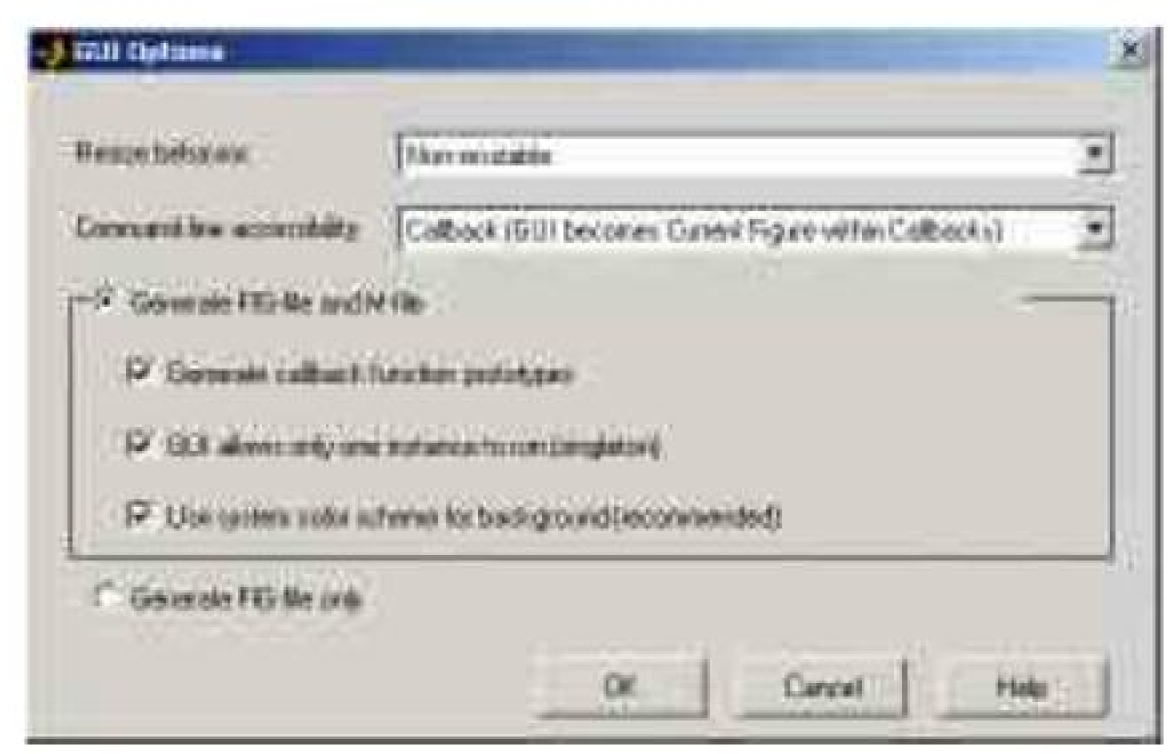


图 7-26　选择生成 M 文件的属性

还可以通过 GUIDE 工具条中的 M 语言编辑器按钮编辑图形界面的 M 文件，这时将首先询问保存 fig 文件，保存 fig 文件后再打开图形界面的 M 语言函数文件。

图形界面的 M 语言函数文件将为不同的控件分别创建至少一个函数，这些函数都作为图形用户界面应用程序的子函数存在，请读者仔细观察由 GUIDE 创建的 M 函数文件，也可以直接编辑本章的例子观察 M 函数文件的内容。

GUIDE 创建的 M 文件一般由调度代码、GUI 回调函数和 GUI 控件回调函数几个不同的部分组成。这里将结合例子说明这几部分代码的作用。

首先，程序的头部为程序的初始化和调度代码，一般情况下，用户不需要修改这部分代码。在程序执行的过程中，这部分代码起到了调度程序的功能，分别完成了打开图形界面、初始化以及响应用户动作的功能。一般地，此段代码如下：

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',   gui_Singleton, ...
                   'gui_OpeningFcn', @simple_gui_OpeningFcn, ...
                   'gui_OutputFcn',   @simple_gui_OutputFcn, ...
                   'gui_LayoutFcn',   [] , ...
                   'gui_Callback',    []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

上面代码中的核心是调用函数 gui_mainfcn，该函数起到了图形界面创建、控件监听等作用，有兴趣的读者可以阅读该函数的代码，加深对图形界面程序创建的认识。

在调度代码的后面紧跟着两个子函数，这两个子函数就是 GUI 的回调函数。第一个回调函数是

```
function simple_gui_OpeningFcn(hObject, eventdata, handles, varargin)
```

该函数负责打开图形界面，同时，若程序中需要对一些全局的参数进行初始化或者设置时，可以将初始化用户数据的代码添加在该子函数中。在该子函数中包含下面一句代码：

```
% Update handles structure
guidata(hObject, handles);
```

这里调用 guidata 函数，将结构 handles 与 GUI 界面共同保存起来。如前文所述，handles 结构中包含了所有图形界面上控件的 Tag 属性值(也就是句柄)，同时还能够完成在不同的回调函数之间共享用户数据的功能。不过，每次修改了 handles 结构内部数据之后，一定要调用 guidata 函数更新该结构的数据，否则在其他的子函数中，就无法使用最新的数据了。

第二个回调函数是

```
function varargout = simple_gui_OutputFcn(hObject, eventdata, handles)
```

该子函数负责将图形界面的句柄返回给用户的输出参数，前提是用户在执行该 M 文件时，在命令行中指定了输出：

```
h = simple_gui;
```

这时 h 将是图形用户界面窗体的句柄。

接下来的子函数是分别用来响应用户的动作输入，完成相应功能的 GUI 控件回调子函数。在这里首先编写 Draw 按钮的回调函数。在 M 文件中找到函数 btnDraw_Callback，并且添加相应的代码：

```
function btnDraw_Callback(hObject, eventdata, handles)
% 绘制三维曲面
hsurfc = surf(peaks(30),'FaceColor','blue');
% 保存三维曲面的句柄
handles.hsurface = hsurfc;
guidata(hObject,handles);
% 设置相应的文本显示当前色彩数值
set(handles.txtRed,'String',['Red: 0' ]);
set(handles.txtGreen,'String',['Green: 0']);
set(handles.txtBlue,'String',['Blue: 1']);
```

在上述的代码中，首先绘制了三维曲面，然后将三维曲面的句柄保存在 handles 结构中。最后还设置了相应色彩的文本属性以显示不同的色彩数值。

注意：再次强调

在 GUIDE 创建的 M 函数文件中，若修改了 handles 结构，则需要通过 guiddata 函数将 handles 的结构保存起来，只有这样才能够通过 handles 结构将不同的用户数据传递到相应的子函数中。有关 guidata 函数请读者参阅 MATLAB 的帮助文档或者在线帮助。

若此时执行 M 文件，单击“Draw”按钮之后，就可以在坐标轴中观察到程序的输出效果——三维的曲面，如图 7-27 所示。

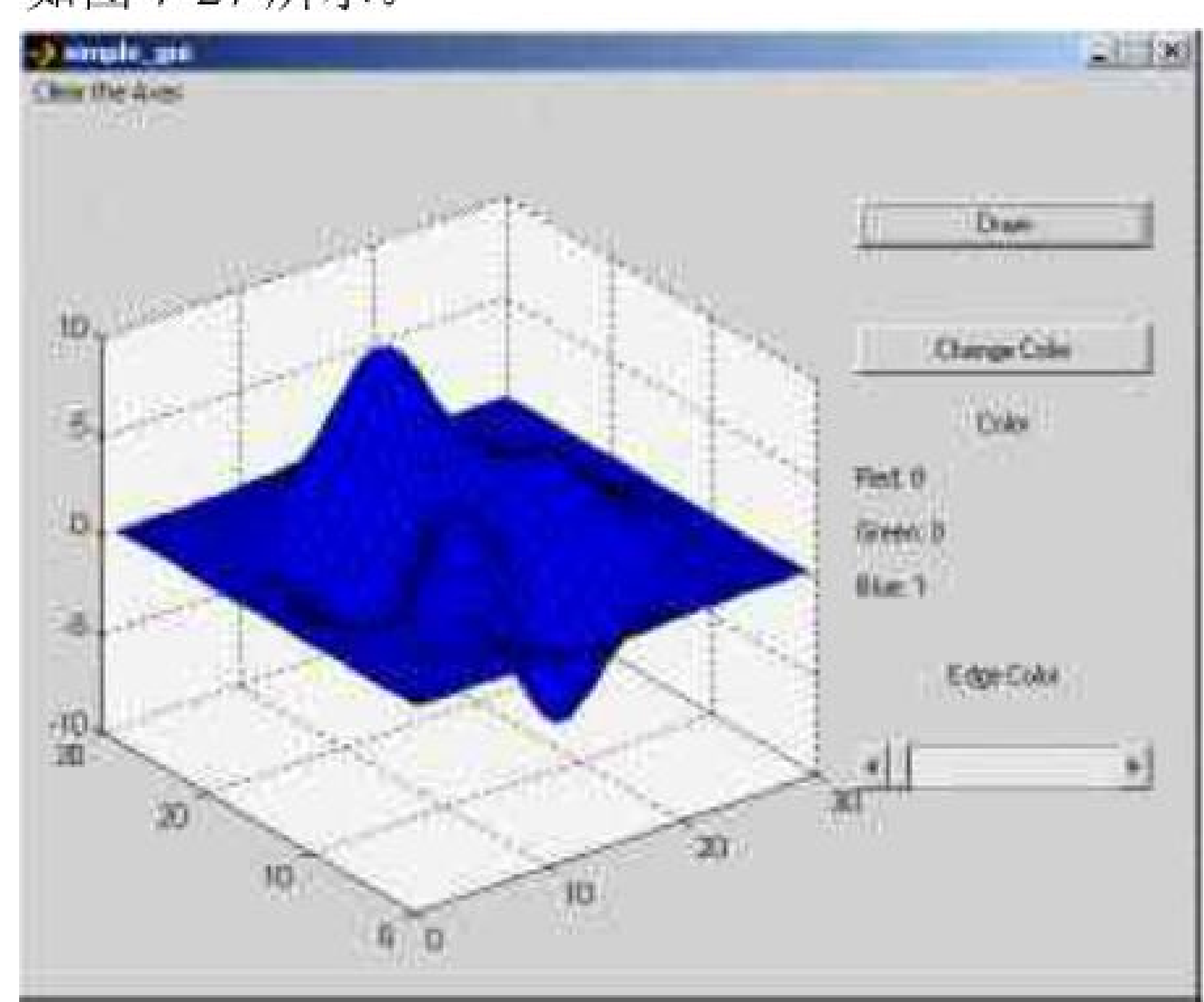


图 7-27　绘制曲面的效果

继续修改 M 文件，在不同控件的回调函数中添加代码完成用户界面的功能。Simple GUI 的 M 代码(回调函数部分)如下：

单击“Change Color”按钮的回调函数：

```
% --- Executes on button press in btnChangeColor.
function btnChangeColor_Callback(hObject, eventdata, handles)
%修改曲面色彩
% 获取曲面的句柄
hsurf = handles.hsurface;
%hsurf = findobj(gcf,'Type','Surface');
% 生成随机的色彩
newColor = rand(1,3);
% 设置曲面的色彩
set(hsurf,'FaceColor',newColor);                    % Set face color of surface
% 设置相应的文本显示当前色彩数值
set(handles.txtRed,'String',['Red: ' num2str(newColor(1))]);
set(handles.txtGreen,'String',['Green: ' num2str(newColor(2))]);
set(handles.txtBlue,'String',['Blue: ' num2str(newColor(3))]);
```

创建滚动条的回调函数：

```
% --- Executes during object creation, after setting all properties.
function sliderEdgeColor_CreateFcn(hObject, eventdata, handles)

usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

滚动条的回调函数：

```
% --- Executes on slider movement.
function sliderEdgeColor_Callback(hObject, eventdata, handles)
% 修改曲面的边缘色彩
% 获取对象句柄
hsurf = handles.hsurface;
%hsurf = findobj(gcf,'Type','Surface');        % Get handle to surface
% 获取滚动条当前的数值
newRed = get(hObject,'Value');                  % Get new color from slider
% 设置新的色彩数值
currentColor= rand(1,3);    % Assign value to first element of HSV
```

```
currentColor(1) = newRed;
% 设置色彩属性
set(hsurf,'EdgeColor',currentColor);
```

菜单命令的回调函数：

```
% --------------------------------------------------------------------
function CleartheAxes_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function ClearAxesDone_Callback(hObject, eventdata, handles)
% 清除当前的坐标轴内容
cla
```

现在图形用户界面的应用程序都编写完毕了，可以运行该 M 文件并且检测相应的功能。图 7-28 为图形用户界面执行过程中的状态之一。

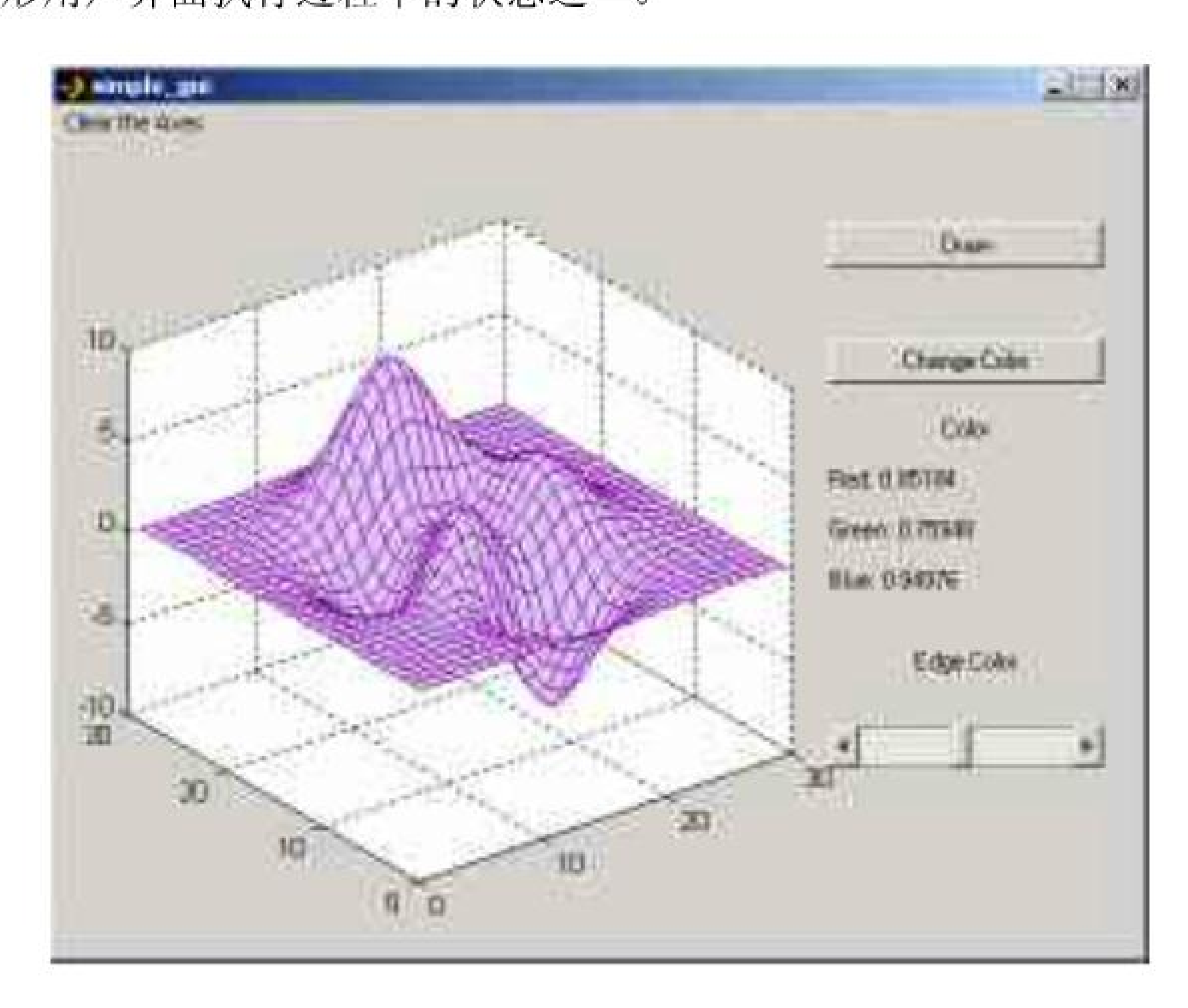


图 7-28 图形用户界面简单例子的执行状态

在这个例子中使用了一个非常重要的函数——guidata，该函数主要用来在图形用户界面中存储或者获取用户数据，它的基本语法为

- 存储数据：guidata(object_handle,data)。
- 获取数据：data = guidata(object_handle)。

这里 object_handle 若不是图形窗体的句柄，就是使用 object_handle 句柄对象的父层次的图形窗体对象句柄来保存数据。guidata 函数为用户提供了一种简便的保存和获取用户应用程序数据的途径，它是在 MATLAB 图形用户界面应用程序中常用的函数之一。

7.6　常用的图形界面函数

为了便于用户创建图形用户界面应用程序，MATLAB 还提供了一些预定义的图形用户界面，比如错误信息对话框、用户输入对话框等，在表 7-2 中，对这些常用的对话框函数进行了总结。

表 **7-2**　常用的图形界面函数

函　数	说　　明
dialog	创建对话框
errordlg	错误信息对话框
warndlg	警告信息对话框
helpdlg	帮助信息对话框
inputdlg	参数输入对话框
listdlg	列表选择对话框
printdlg	打印图形对话框
pagesetupdlg	打印页面设置对话框
questdlg	询问对话框
msgbox	信息显示对话框
uiload	加载数据文件的标准图形界面
uisave	保存数据文件的标准图形界面
uigetdir	选取目录的标准图形界面
uigetfile	选取文件的标准图形界面
uiputfile	保存文件的标准图形界面
uisetcolor	设置色彩的标准图形界面
uisetfont	设置字体的标准图形界面
waitbar	显示包含进度条的对话框

7.7　本 章 小 结

在本章简要讨论了在 MATLAB 中创建图形用户界面的方法，详细介绍了 MATLAB 中图形对象的层次以及图形对象句柄的使用方法，以及 MATLAB 创建图形用户界面应用程序的集成开发环境 GUIDE 的使用方法。本章通过一个简单的示例说明了创建图形用户界面的过程。在 MATLAB 中创建图形用户界面有两种方法，本章介绍的 GUIDE 是相对比较简单、易行的方法，而另外一种方法——图形句柄，相对的编程过程比较繁琐，工作量也很大，所以有兴趣的读者请参阅 MATLAB 的帮助文档。另外，尽管能够利用 M 语言创建图形用户界面，希望读者依然牢记一点，就是 MATLAB 的图形用户界面能力比较有限，它不能像 VC 或者 VB 那样创建出具有 Windows 风格的图形窗体，也不能像 Java 或者 Python 语言那样创建出功能复杂的用户界面。当用户的需求难以使用 M 语言实现的时候，可以考虑使用 Java 语言来实现相应的功能。有关在 M 语言中使用 Java 语言的相关知识请参阅 MATLAB 的帮助文档或者《MATLAB 外部接口编程》一书。

附录 A MATLAB 关键字

MATLAB 的关键字可以通过在 MATLAB 命令行窗口中键入下面的指令获取：

```
>> iskeyword
ans =

    'break'
    'case'
    'catch'
    'continue'
    'else'
    'elseif'
    'end'
    'for'
    'function'
    'global'
    'if'
    'otherwise'
    'persistent'
    'return'
    'switch'
    'try'
    'while'
```

以上就是 MATLAB 的关键字或者叫作保留字，在选择函数或者变量名称的时候，请不要使用这些字符作为变量或者函数的名称。

附录 B　MATLAB 可用的 LaTeX 字符集

标识符	符号	标识符	符号	标识符	符号
\alpha	α	\upsilon	υ	\sim	~
\beta	β	\phi	ϕ	\leq	≤
\gamma	γ	\chi	χ	\infty	∞
\delta	δ	\psi	ψ	\clubsuit	♣
\epsilon	ε	\omega	ω	\diamondsuit	♦
\zeta	ζ	\Gamma	Γ	\heartsuit	♥
\eta	η	\Delta	Δ	\spadesuit	♠
\theta	θ	\Theta	Θ	\leftrightarrow	↔
\vartheta	ϑ	\Lambda	Λ	\leftarrow	←
\iota	ι	\Xi	Ξ	\uparrow	↑
\kappa	κ	\Pi	Π	\rightarrow	→
\lambda	λ	\Sigma	Σ	\downarrow	↓
\mu	µ	\Upsilon	ϒ	\circ	°
\nu	ν	\Phi	Φ	\pm	±
\xi	ξ	\Psi	Ψ	\geq	≥
\pi	π	\Omega	Ω	\propto	∝
\rho	ρ	\forall	∀	\partial	∂
\sigma	σ	\exists	∃	\bullet	•
\varsigma	ς	\ni	∍	\div	÷
\tau	τ	\cong	≅	\neq	≠
\equiv	≡	\approx	≈	\aleph	ℵ
\Im	ℑ	\Re	ℜ	\wp	℘
\otimes	⊗	\oplus	⊕	\oslash	Ø
\cap	∩	\cup	∪	\supseteq	⊇
\supset	⊃	\subseteq	⊆	\subset	⊂
\int	∫	\in	∈	\o	°
\rfloor	⌋	\lceil	⌈	\nabla	∇
\lfloor	⌊	\cdot	·	\ldots	...
\perp	⊥	\neg	¬	\prime	·
\wedge	∧	\times	×	\0	∅
\rceil	⌉	\surd	√	\mid	∣
\vee	∨	\varpi	ϖ	\copyright	©
\langle	〈	\rangle	〉		

附录 C　MATLAB 的安装

C.1　Windows 系统下的安装

在 Windows 系统下安装 MATLAB 的过程非常简便，在这里需要首先强调一下安装 MATLAB 产品的基本配置：

- CPU：Pentium、Pentium Pro、Pentium II、Pentium III、Pentium IV、Intel Xeon、AMD Athlon 或 Athlon XP。
- 支持的操作系统：Microsoft Windows 98 、 Windows Millennium Edition (ME)、Windows NT 4.0 (with Service Pack 5 for Y2K compliancy 或 Service Pack 6a)、Windows 2000、Windows XP。
- CD-ROM drive (for installation from CD)。
- 至少 128 MB RAM，推荐使用 256 MB RAM。
- 硬盘空间至少需要 120MB 用于安装 MATLAB 产品，若安装 MATLAB 的帮助文档则需要 260MB 硬盘空间，如果安装其他工具箱，则需要的硬盘空间更大。
- 推荐使用 16、24 或 32 位支持 OpenGL 的图形加速卡。

其他的需求：

- 推荐安装打印机、声卡、AGP 显卡。
- Microsoft Word 8.0 (Office 97)、Office 2000 或 Office XP。
- 安装 TCP/IP 协议。
- 安装 FLEXlm 8.0d（由 MathWorks 提供的产品安装程序提供）。

推荐安装下列编译器之一：

- Compaq Visual Fortran 5.0、6.1 或 6.5。
- Microsoft Visual C/C++ version 5.0、6.0 或 7.0。
- Borland C/C++ version 5.0 or 5.02。
- Borland C++ Builder version 3.0、4.0、5.0 或 6.0。
- WATCOM version 10.6 or 11。
- Lcc 2.4 (随 MATLAB 发布)。

在安装产品之前，需要确认如下的工作：

- 请退出所有已经运行的应用程序，特别是计算机中安装有实时病毒防火墙时，请暂时关闭病毒防火墙。
- 如果在 Windows NT、Windows 2000 或者 Windows XP 系统下安装，则需要获得系统管理员的身份。

■ 请用户确认自己是否拥有合法的 MATLAB 产品的 PLP——Personal License Password。

将 Mathworks 公司提供的产品光盘放置于计算机的 CD-ROM 中，一般地，Windows 将执行产品光盘中的 Autorun 程序，否则，请执行光盘上的 Setup.exe 文件，这时将进入 MATLAB 的安装程序，如图 C-1 所示。



图 C-1　MATLAB 应用程序的欢迎对话框

单击对话框的“Next”按钮，将进入输入 PLP 信息的对话框，如图 C-2 所示。



图 C-2　输入 PLP 信息的对话框

一般地，PLP——Personal License Password 信息是在用户购买了 MATLAB 的产品之后，由 Mathworks 公司通过 E-mail 向客户发送的最终用户信息，其中对于 Windows 平台下的用户，一般文件中会包含一串以数字 13 开头的代码，例如，13-12345-12345-12345……。这串数字就是需要在如图 C-2 的对话框中输入的信息，可以直接从 E-mail 文件中将此串数字拷贝粘贴到对话框中，然后单击“Next”按钮继续安装。接着将显示 Licencse Agreement、Personal Information 等对话框，请用户在相应的对话框中输入自己的用户信息，如图 C-3 所示。

一路单击“Next”按钮继续安装之后，将进入产品列表（Product List）对话框，在该对话框中需要用户确认是否安装帮助文档、MATLAB 产品的安装路径以及需要安装的 MATLAB 产品模块等信息，如图 C-4 所示。

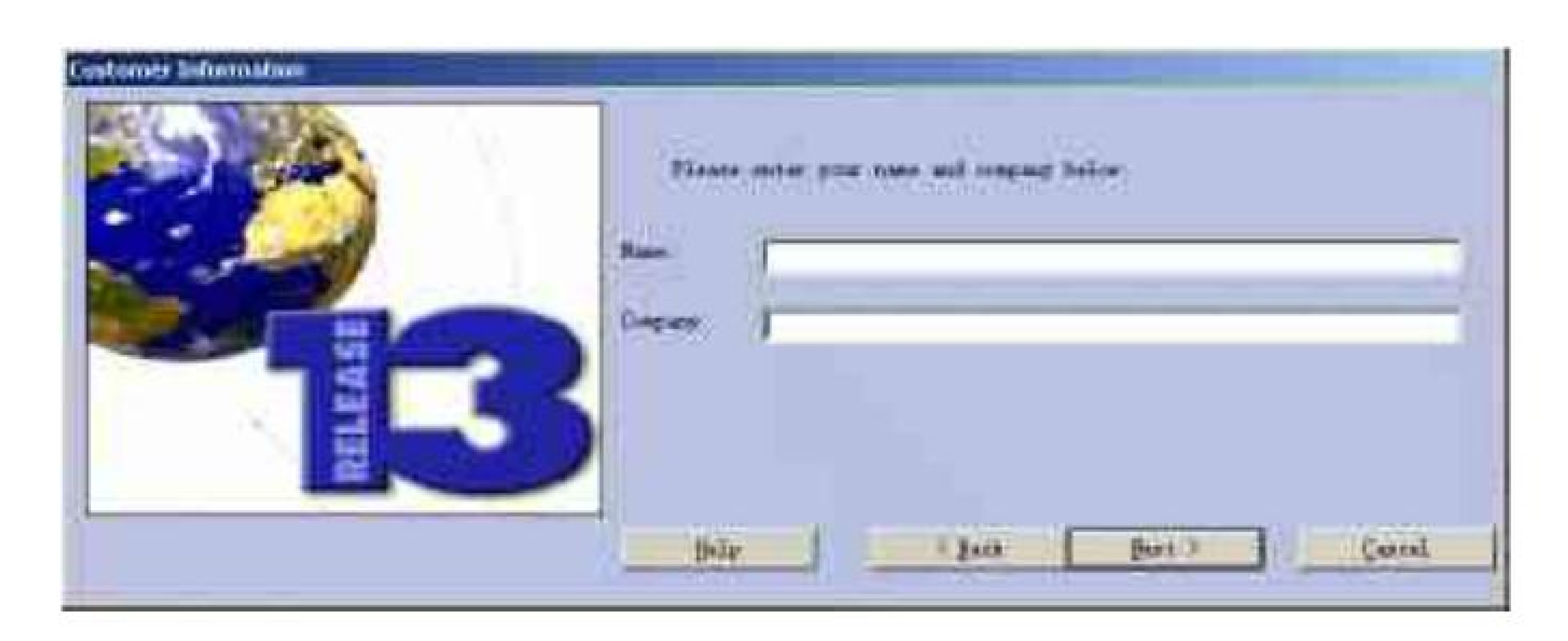


图 C-3 用户信息对话框

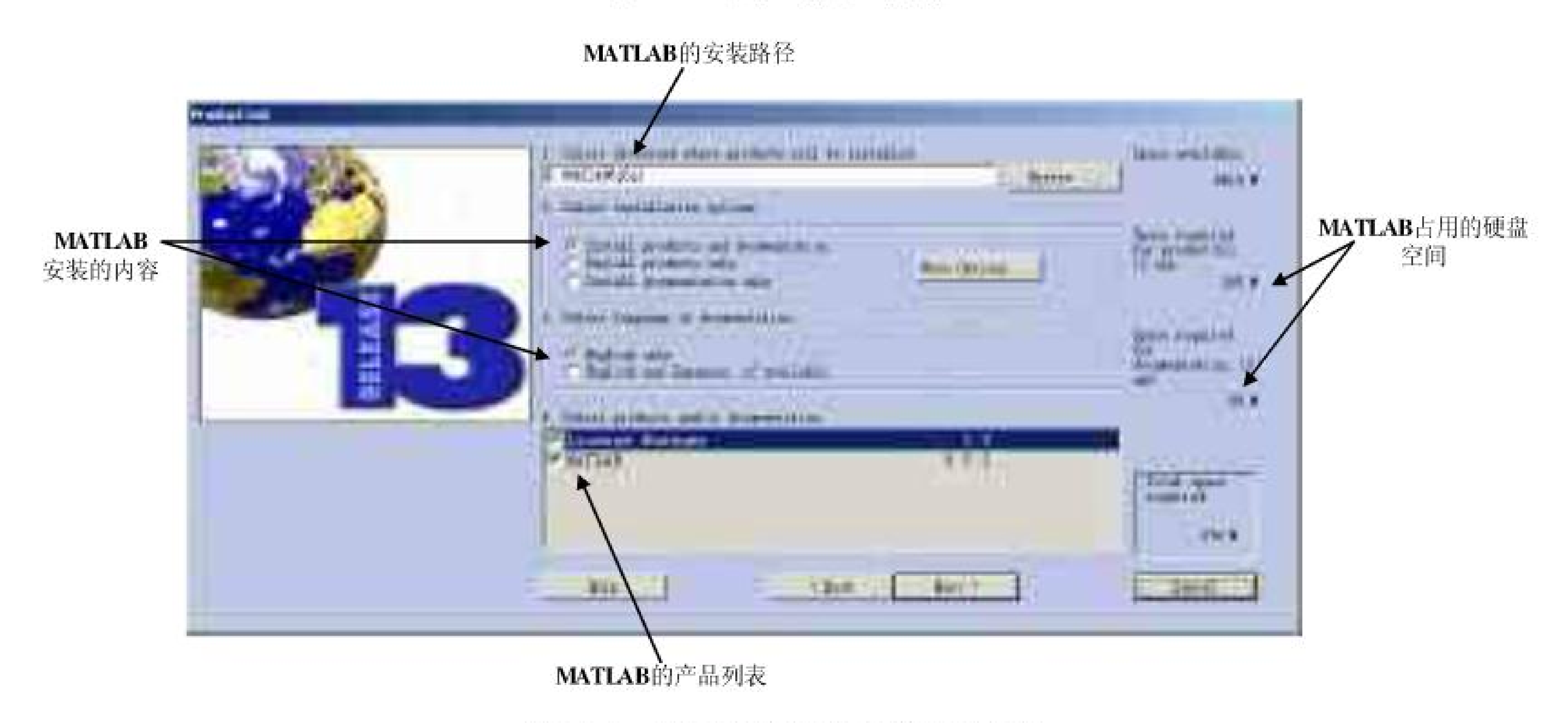


图 C-4 选择产品的安装列表等信息

单击“Next”按钮，将进入 MATLAB 的安装过程，安装所消耗的时间和用户安装 MATLAB 的计算机性能以及选择的 MATLAB 产品模块数量有关。完成安装之后，如果系统不需要重新启动计算机，则安装程序显示如图 C-5 所示的对话框。



图 C-5 完成安装——启动 MATLAB

如果用户安装的产品模块需要重新启动计算机，则显示如图 C-6 所示的对话框。

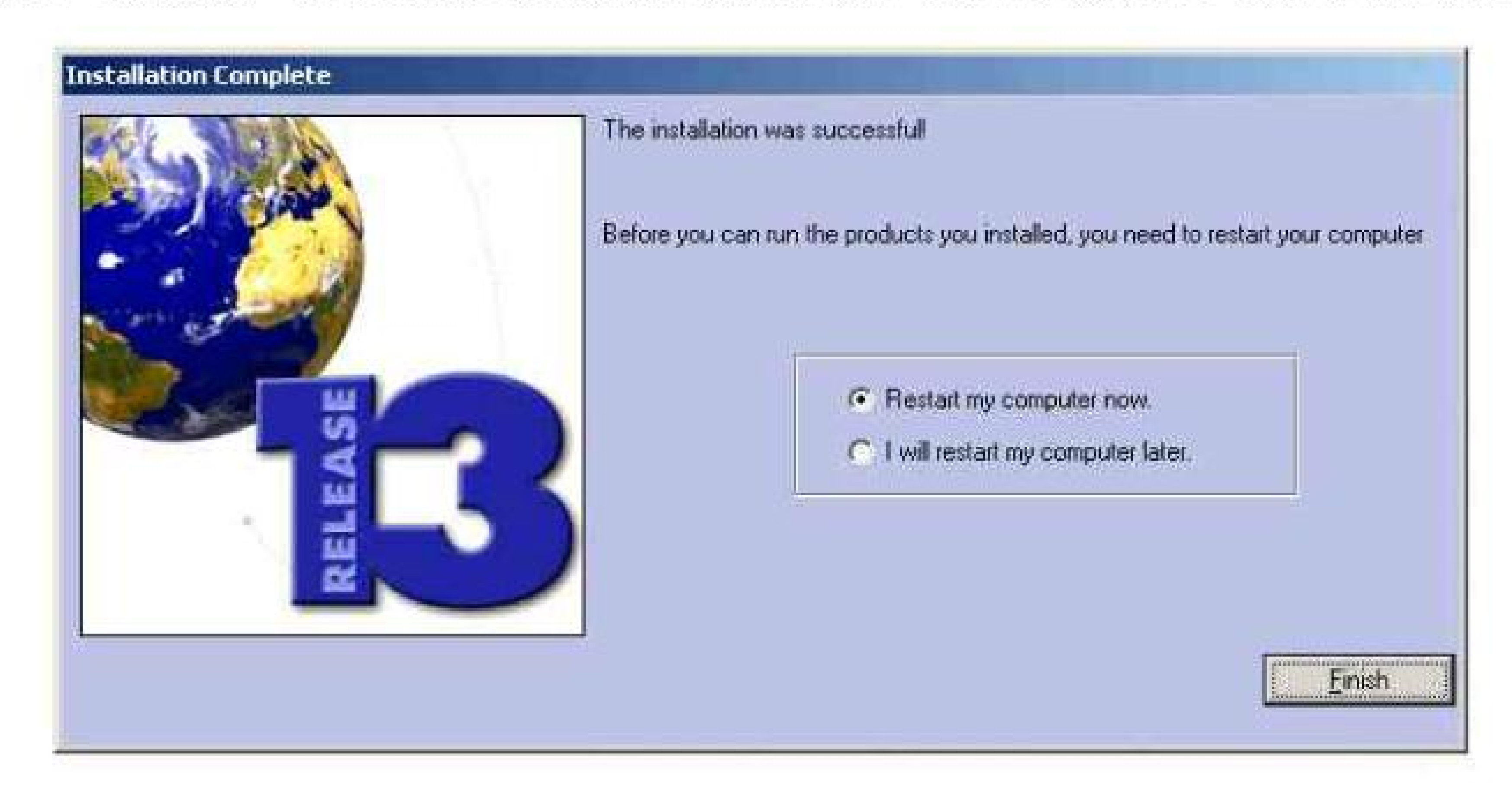


图 C-6　完成安装——重新启动计算机

请用户根据出现的不同对话框，选择需要执行的工作。

在安装完毕之后，MATLAB 安装程序将在桌面上创建 MATLAB 应用程序快捷方式，双击该快捷方式，或者执行“开始”菜单下的相关命令就可以启动 MATALAB 了。默认的 MATLAB 启动之后将使用%MATLABROOT%\Work 路径作为当前的工作路径，如果需要修改，则可以在 MATLAB 的快捷方式上单击鼠标右键，执行“属性”命令，在弹出的对话框中，设置快捷方式应用程序的起始位置为自己需要的路径，例如 D:\Temp，如图 C-7 所示。



图 C-7　设置起始位置

C.2 Unix 系统下的安装

MATLAB 产品不仅仅可以安装在 Windows 操作系统下，而且还可以安装在 Unix 操作系统下，目前 MATLAB Release 13 版本支持的 Unix 操作系统见表 C-1。

表 **C-1 MATLAB** 支持的 **Unix** 操作系统

开发商	系统硬件	操作系统	版本
Sun	SPARC， ULTRA	Solaris	2.6、 2.7、 2.8
HP	PA-RISC 1.1	HP-UX	10.20
HP	PA-RISC 2.0	HP-UX	11.0
SGI	R5000、R8000、R10000、R12000	IRIX, IRIX 64	6.5.6～6.5.12
Compaq	Alpha	Tru64 UNIX	4.0f、 5.0、 5.1
IBM	RS/6000	AIX	4.3.3、 5.1
Linux	Pentium, Pentium Pro、II、III、IV， Intel Xeon， AMD Athlon 或 Athlon XP	ix86-Linux X-Window(X11R6)	2.2.x、 2.4.x Kernal

不同的操作系统下的安装步骤可能不一致，下面以 Solaris 操作系统 2.8 版本为例，说明 MATLAB 在 Unix 操作系统下的安装过程。

首先需要使用 ROOT 身份登录操作系统，并且将 MATLAB 的第一张安装光盘，放入 CD-ROM 中，然后在操作系统中加载光驱。

进入安装产品的路径：

```
cd /usr/local
mkdir matlab6p5 (仅首次安装时需要此步骤)
cd matlab6p5
```

然后将 License 文件拷贝到安装路径中。安装 MATLAB 需要的 License 文件需要按照 Mathworks 提供给用户的产品信息文件创建，正式购买了 Unix 版本的 MATLAB 的用户将接收到包含产品信息的 E-mail，将 E-mail 文件中以#BEGIN 开头到#END 结尾的内容拷贝到新的文件中，一般内容如下：

```
# BEGIN-------cut here-------CUT HERE-------BEGIN
# MATLAB license passcode file for use with FLEXlm.
# LicenseNo: 12345              HostID: ID=12345
INCREMENT TMW_Archive MLM 13 01-jan-0000 0 9CC470AGGCCB4A810 \
  VENDOR_STRING="f" HOSTID=DEMO SN=12345
INCREMENT MATLAB MLM 13 01-jan-0000 1 BCECD70AD686BGG7E9917 \
  USER_BASED DUP_GROUP=UH SN=12345
INCREMENT SIMULINK MLM 13 01-jan-0000 1 2C5C2G75GF38BGGEAD8FD \
  USER_BASED DUP_GROUP=UH SN=12345
INCREMENT Control_Toolbox MLM 13 01-jan-0000 1 \
  7CACA7GA23GG5C5E38C1 USER_BASED DUP_GROUP=UH SN=12345
INCREMENT Identification_Toolbox MLM 13 01-jan-0000 1 \
```

EC5C175AGG025G2AD73A USER_BASED DUP_GROUP=UH SN=12345

END---------cut here-------CUT HERE-------END

并且将该文件命名为 license.dat，安装 MATLAB 的时候就需要此文件。

执行光盘上的安装脚本——/cdrom/install* &。执行脚本之后，将出现欢迎对话框、产品安装路径设置对话框、Lisence 文件对话框，接着出现的就是产品列表对话框，如图 C-8 所示。

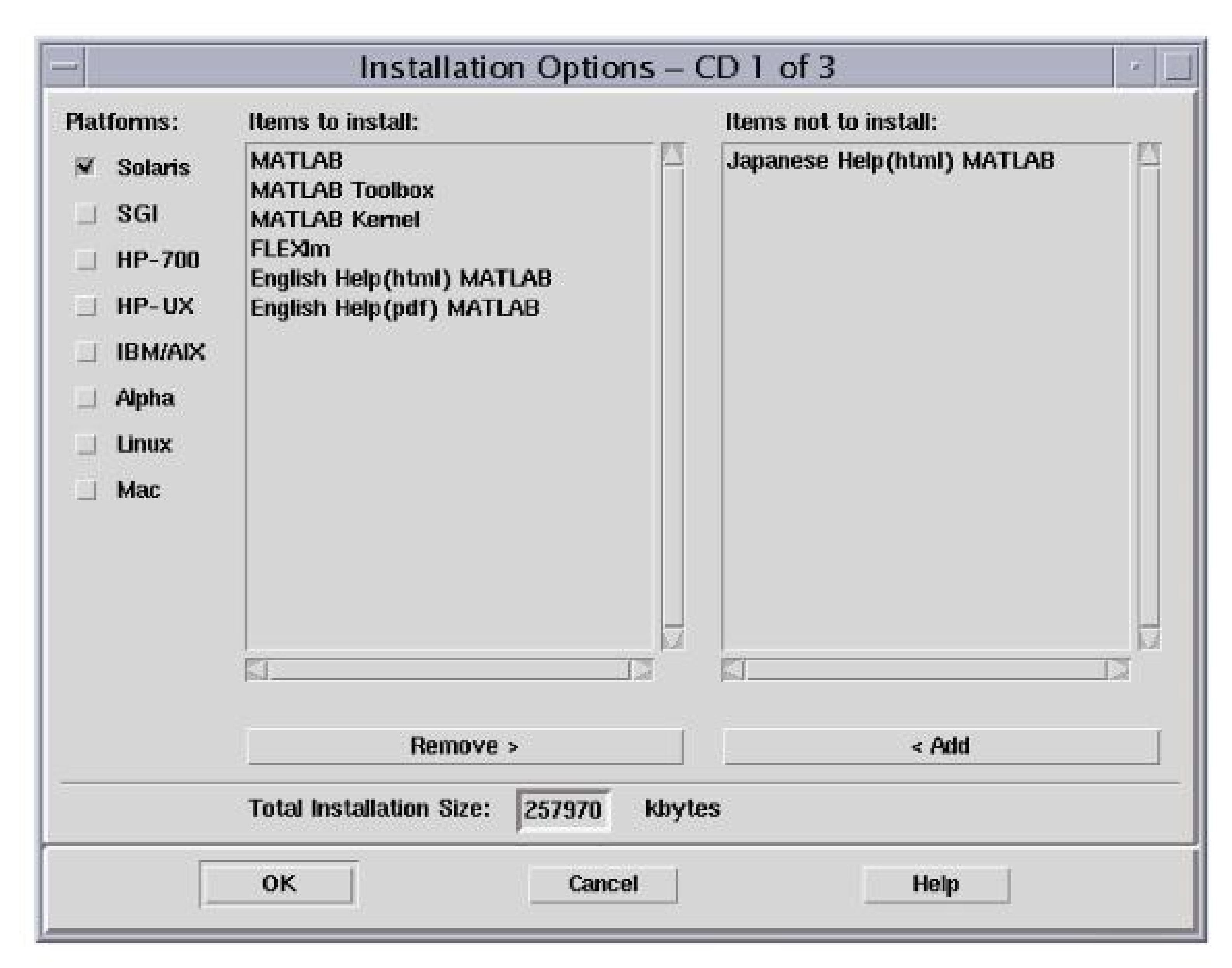


图 C-8　产品列表对话框

这时，用户可以通过对话框中的“Remove”按钮或者“Add”按钮删除、添加需要安装到操作系统中的产品模块，确认之后，单击“OK”按钮继续产品的安装。

不过需要说明一点，在 Unix 操作系统中安装 MATLAB 产品时，需要重复上述的安装步骤，因为 MATLAB 的产品光盘一共有三张，每次安装的时候都需要执行相应光盘上的安装脚本，这样才能够完成所有产品的安装。用户能够安装的产品模块，则由 Mathworks 公司提供给用户的 License 文件的内容决定。

安装完毕之后，在 MATLAB 的安装路径下执行 matlab 指令就可以启动 MATLAB 产品了。

注意：

由于 Unix 产品的多样性，每一种 Unix 操作系统的操作方法和安装过程可能都有所不同，请用户安装 MATLAB 的时候仔细阅读 MATLAB 的安装手册，避免在安装 MATLAB 的时候出现不必要麻烦。

另外，MATLAB 还可以安装在 MAC 系统上，这里就不再赘述了，有兴趣的读者请参阅 MATLAB 的帮助文档。

附录 D 北京九州恒润科技有限公司简介

北京恒润科技有限公司成立于 1998 年 4 月，是一家民营高新技术企业。公司主要业务涉及工程系统设计仿真产品的代理销售、工程仿真分析项目咨询等。现有员工 53 人，其中专家三人，博士五人，硕士十五人。

公司成立以来，承揽了众多的工程咨询项目，为用户提供了全方位的技术解决方案和技术支持，具备了很强的软件开发和技术咨询实力。目前公司项目咨询领域主要有导航、制导与控制，射频仿真和汽车电子。

公司注重建设具有特色的公司文化，培养员工认同共同的价值观，确保员工能伴随着公司的发展而发展，从而形成了激励上进型的工作氛围，建立了一支高素质的、有凝聚力的队伍。与此同时，公司管理层不断借鉴成功公司的组织体系和管理经验，经过消化、试行、改进、完善的过程，构建了公司特有的组织与管理体系、技术研发与服务体系、市场开拓与支持体系，目前公司已开始进入一个高速发展期。

公司在上海、成都两地设立了办事处，并于 2003 年 3 月被授予为北京市首批“纳税信用 A 级企业”。

公司的文化和价值取向：诚信，以人为本，追求卓越。

公司的发展方向：以为用户提供集成工程环境，提供技术解决方案为依托，进行自主产品的研制、开发。

公司当前的产品信息

公司随时追踪国内外技术发展的最新动态，把先进技术和一流产品介绍给国内用户，代理了若干处于国际领先的仿真分析和设计的工具产品，支持用户进行工程系统设计和仿真分析。这些产品包括：

? 美国 MathWorks 公司的 MATLAB 产品（独家代理）。
? 德国 dSPACE 公司实时半实物仿真系统 dSAPCE（独家代理）。
? 德国 Vector 公司 CAN 网络系统的分析工具 Vector(独家代理)。
? 英国 Radioscape 公司的通讯仿真软件 RadioScape（独家代理）。
? 美国 Signalogic 公司的 DSP 设计的辅助软件 Signalogic（独家代理）。
? 德国 TESIS 公司的 TESIS 虚拟车辆实时仿真专家软件（独家代理）。
? 美国 PSS 公司的 PSS 卫星控制系统设计分析和仿真软件（独家代理）。
? 美国 TAC 公司的 NEVADA 热辐射分析软件（独家代理）。
? 美国 Network Analysis 公司的 SINDA/G 热分析（独家代理）。
? 美国 Altia 公司的 Altia 嵌入系统图形界面开发平台（独家代理）。
? 英国 Ricardio 公司的车辆发动机、内燃机设计分析系统 Ricardio (独家代理)。

详细的产品信息请登录北京九州恒润科技有限公司网站 www.hirain.com。

培训服务

作为 Mathworks 公司中国大陆地区的独家代理，北京九州恒润科技有限公司拥有完善、权威的 MATLAB 培训服务，并且拥有丰富的培训经验和一流的培训讲师。欢迎使用 MATLAB 从事系统分析、仿真、设计的工程人员、大学教师和学生参加由北京九州恒润科技有限公司提供的 MATLAB 培训课程。请登录公司网站——www.hirain.com 查阅详细的培训信息。

基础培训：

- MATLAB 基础和编程入门
- Simulink 建立动态系统模型
- Real-Time Workshop 基础
- Stateflow 建模技术基础

中级培训：

- MATLAB 外部接口编程
- MATLAB 高级编程技巧
- MATLAB 图形用户界面
- Simulink 高级建模技术
- Stateflow 高级建模技术

高级培训：

- 根据客户特定需要定制

联系方式

北京九州恒润科技有限公司
www.hirain.com
北京公司总部
北京市西城区北三环中路 27 号商房大厦 430 室
电话：010-82011456
上海办事处
上海市徐汇区漕宝路 70 号光大会展中心 D 座 505 室
电话：021-64325416
成都办事处
成都市人民南路一段 86 号城市之心大厦 23 楼 N 座
电话：028-86203381/2/3

参 考 文 献

[1] MATLAB Programming Tips Version 6. Mathworks Inc., September 2003
[2] Using MATLAB Version 6. Mathworks. Inc., September 2003
[3] Using MATLAB Graphics. Mathworks. Inc., September 2003
[4] Creating Graphical User Interfaces. Mathworks Inc., September 2003
[5] Michael Robbins. Good MATLAB Programming Practices for the Non-Programmer
[6] Richard Johnson. MATLAB Programming Styles Guidelines USA Datatool Version 1.5
[7] 唐发根，刘又诚. 数据结构教程. 北京：北京航空航天大学出版社，1996